МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ
ХАРЬКОВСКИЙ НАЦИОНАЛЬНЫЙ УНИВЕРСИТЕТ
РАДИОЭЛЕКТРОНИКИ

В.М. Левыкин, М.В. Евланов, М.А. Керносов

ПАТТЕРНЫ ПРОЕКТИРОВАНИЯ ТРЕБОВАНИЙ К ИНФОРМАЦИОННЫМ СИСТЕМАМ: МОДЕЛИРОВАНИЕ И ПРИМЕНЕНИЕ

Монография

Харьков 2014

УДК 65.011.56:519.876.5:004.896
ББК
Л ХХ

Левикін В.М., Євланов М.В., Керносов М.А. Паттерни проектування вимог до інформаційних систем: моделювання та використання: монографія. – Харків: ХНУРЕ, 2014. – 320 с. – Рос. мовою.

Монографія містить аналіз, постановку задачі та авторське вирішення проблеми повторного використання вимог, реалізованих під час виконання ІТ-проектів створення інформаційних систем різного призначення. Запропоновано нову концепцію подання вимог до інформаційної системи та сформульовано визначення паттерна проектування вимог. Розроблені моделі паттернів проектування вимог до інформаційної системи на рівнях інформації, даних та знань. Розглянуто основні особливості архітектурного фреймворка та інтелектуальної інформаційної технології прискореної розробки інформаційних систем, що засновані на наведених результатах. Проведено оцінювання ефективності використання розглянутої технології в рамках ІТ-проекту створення різних інформаційних систем.

Призначено для наукових працівників, аспірантів і студентів напрямку «Комп'ютерні науки».

Іл.: 49, Табл. 22. Бібліогр. наймен. 154.

Монография содержит анализ, постановку задачи и авторское решение проблемы повторного использования требований, реализованных в ходе выполнения ИТ-проектов создания информационных систем различного назначения. Предложена новая концепция представления требований к информационной системе и сформулировано определение паттерна проектирования требований. Разработаны модели паттернов проектирования требований к информационной системе на уровнях информации, данных и знаний. Рассмотрены основные особенности архитектурного фреймворка и интеллектуальной информационной технологии ускоренной разработки информационных систем, основанных на приведенных результатах. Проведена оценка эффективности применения рассмотренной технологии в рамках ИТ-проектов создания различных информационных систем.

Предназначено для научных работников, аспирантов и студентов направления «Компьютерные науки».

Ил.: 49, Табл. 22. Библиогр. наим. 154.

Рецензенти:
- Годлевський М.Д., д-р техн. наук, професор, зав. каф. автоматизованих систем управления НТУ «ХПІ», м. Харків;
- Федорович О.Є., д-р техн. наук, професор, зав. каф. інформаційних управляючих систем НАУ «ХАІ», м. Харків;
- Асєєв Ю.Г., д-р техн. наук, професор, зав. каф. інформаційних технологій ХДАК, м. Харків.

СОДЕРЖАНИЕ

ПРЕДИСЛОВИЕ

Разработка современных информационных систем (ИС) характеризуется, прежде всего, стандартизацией практических действий, являющихся наиболее удачными в рамках отдельных проектов их создания (ИТ-проектов). Анализ и обобщение этих практик позволяют выделить наборы рекомендаций относительно процессов и действий, выполнение которых позволяет переходить из одной стадии жизненного цикла ИС в другую. В настоящее время такие наборы практик сведены в стандарты серии ISO (или их аналоги в отдельных странах), устанавливающие общие принципы, подходы и правила выполнения работ по созданию, внедрению, эксплуатации и развитию ИС и программных систем различного назначения.

Главной особенностью данных стандартов по отношению к практикам, определяющим процессы создания систем и, в частности, ИС, является представление создаваемой системы в виде совокупности описаний. Эта совокупность описаний по мере выполнения процессов создания ИС в рамках ее жизненного цикла трансформируется от представления системы в виде «черного ящика» к представлению той же системы в виде сначала «серого ящика», а впоследствии – «белого ящика». При этом сами процессы, зафиксированные в современных стандартах, можно рассматривать как работы и узлы принятия решений, снижающие уровень неопределенности описаний создаваемой системы.

Такое представление ИС заставляет обратить особое внимание на процессы ее создания, формирующие самые первые описания системы – описания, представляющие ИС в виде совокупности требований к ней. Именно эти описания устанавливают основу для общесистемных решений и решений по отдельным видам обеспечений, а ошибки, допускаемые в процессах формирования таких описаний, являются наиболее дорогостоящими ошибками любого ИТ-проекта. В дальнейшем такие процессы будем называть процессами макропроектирования ИС.

IDEF3-модель процессов макропроектирования ИС, формирующих и анализирующих совокупность требований к системе, а также синтезирующих набор описаний системы, связывающих выдвинутые к ней требования с решениями по отдельным видам обеспечений, показана на рисунке. Данная модель соответствует действующему в настоящее время стандарту ISO 15288:2002, устанавливающему описания процессов жизненного цикла системы и, в частности, информационной системы.

По результатам анализа приведенной на рисунке модели можно сделать следующие выводы:

а) в ходе выполнения процессов формируется совокупность отдельных требований к ИС, однако анализ взаимосвязей этих требований производится только в процессе проектирования архитектуры;

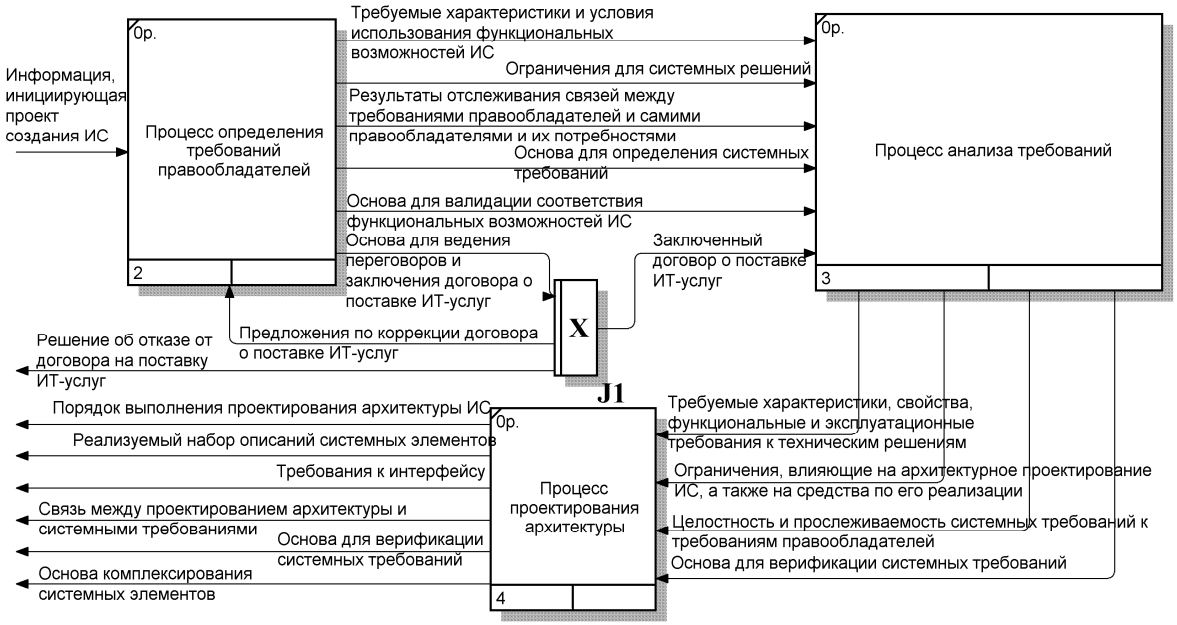

Рисунок – IDEF3-модель взаимосвязей процессов макропроектирования информационной системы

б) показанные на рисунке в виде логического перекрестка «исключающее ИЛИ» (J1) действия по принятию решения о заключении договора на поставку ИТ-услуг и, соответственно, решения о продолжении работ по созданию ИС, основываются на информации о совокупности отдельных требований к ИС и не учитывают взаимосвязь этих требований. В результате этого открывается возможность принятия ошибочного решения о поставке ИТ-услуг, которые в своей совокупности не дают ожидаемого эффекта от своей эксплуатации;

в) показанные на рисунке процессы не учитывают опыт создания ИС аналогичного назначения, накопленный организацией, выполняющей эти процессы, в предыдущих ИТ-проектах, хотя использование такого опыта значительно сокращает время и затраты других ресурсов на создание конкретной ИС.

Следует отметить, что вопросам представления и анализа требований к программным системам и, в частности, требований к ИС посвящены труды целого ряда исследователей. Среди этих исследователей следует упомянуть Б. Боэма, Д. Леффингуэлла, Д. Уидрига, А. Кобёрна. К.И. Виггерса и других. Не менее серьезное внимание уделяется вопросам анализа и проектирования системной архитектуры в трудах Дж. Захмана, П. Хелланд и целого ряда современных исследователей, участвующих в разработке и совершенствования стандартов серии ISO. Не менее важным являются работы коллективов

большинства крупных организаций, занятых разработкой ИС и программных систем, посвященные прикладным аспектам проблемы создания систем и технологий управления требованиями.

В Украине решению вопросов разработки и моделирования сложных ИС, в том числе ИС управления организационно-техническими объектами, посвящены работы исследователей, выполненные под руководством А.А. Павлова и В.И. Гриценко. В этих работах рассматриваются теоретические основы и прикладные технологии моделирования и проектирования ИС различного назначения. Проблема управления предприятием как совокупностью интеллектуальных производственных систем рассмотрена в работах В.В. Казимира. Созданию архитектур, моделей, методов и информационных технологий адаптивной разработки и реинжиниринга информационно-управляющих систем для организационно-технических объектов посвящены работы Н.В. Ткачука. Проблемы онтологического моделирования ИС и процессов их разработки рассмотрены в работах, выполненных под руководством А.В. Палагина. В этих работах рассматриваются вопросы, связанные с интеллектуализацией разработки и сопровождения ИС. Разрабатываемые модели, методы и средства позволяют реализовать вопросы создания знание-ориентированных ИС, особенности которых определяются онтологическими моделями предметной области и их изменениями.

Однако большинство этих исследований оставляют открытым вопрос применимости опыта, накопленного в ходе выполнения предыдущих ИТ-проектов создания ИС, при формировании и анализе требований к создаваемой ИС. Поэтому можно утверждать, что работы, посвященные созданию новых и усовершенствованию существующих моделей, методов и инструментальных средств, позволяющих повысить эффективность и снизить неопределенность выполнения процессов макропроектирования ИС, в том числе – за счет повторного использования требований к созданным ранее системам в ходе выполнения ИТ-проекта создания новой ИС, являются весьма актуальными с теоретической и, особенно, с практической точек зрения.

Предлагаемая вашему вниманию монография посвящена одному из возможных решений проблемы повторного использования требований к ИС в ходе создания новой системы. В монографии рассмотрены основные проблемы макропроектрирования ИС, современные представления требований к ИС и их классификации (раздел 1), обосновано применение сервисного подхода к созданию современных ИС, сформулированы основные определения понятия «требование к ИС» и приведены обобщенные модели этих требований (раздел 2), сформулирована новая концепция представления требований к ИС и предложена модель множества интеллектуальных ИТ формирования и анализа требований к ИС (раздел 3), разработаны модели паттернов представлений требований к ИС на различных уровнях (раздел 4) и рассмотрены основные

аспекты информационной технологии, реализующей заявленные теоретические результаты (раздел 5).

Данная монография может быть полезна исследователям, занимающимся решением теоретико-прикладных вопросов создания ИС и программных систем различного назначения, а также инженерам-практикам, специализирующимся на организации и управлении ИТ-проектами создания новых ИС.

Авторы выражают благодарность сотрудникам и студентам кафедры информационных управляющих систем ХНУРЭ, принимавшим участие в создании монографии, а особенно:

– к.т.н., доц. Васильцовой Наталье Владимировне за ценные рекомендации, помощь и поддержку, оказанные в ходе разработки моделей паттернов проектирования требований к ИС;

– Керносовой Марине Эдуардовне за активное участие в апробации интеллектуальной ИТ ускоренной разработки ИС в ходе создания информационной системы «Реестр Лиги украинских клубов интеллектуальных игр», а также за понимание и терпение.

Отдельную благодарность авторы выражают первому читателю монографии – к.т.н., в.н.с. кафедры информационных управляющих систем ХНУРЭ Неумывакиной Ольге Евгеньевне – за ценные рекомендации, внимание и разностороннюю эрудицию, неоднократно проявленные в ходе создания монографии.

1 АНАЛИЗ ПРОБЛЕМ МАКРОПРОЕКТИРОВАНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ УПРАВЛЕНИЯ ПРЕДПРИЯТИЕМ

1.1 Информационные системы: основные определения

В настоящее время термин «информационная система» (ИС) является весьма многозначным. Можно сказать, что с начала 2000-х гг. вместо единой точки зрения на семантику данного термина стали преобладать множество различных его определений, отличия которых обусловлены целями формулирования этих определений. Данная ситуация объясняется, прежде всего, сложностью современных ИС, которые, как правило, не могут быть описаны единым определением, наиболее полно характеризующим их со всех важных точек зрения одновременно. Основное внимание в таких определениях сосредоточено на тех аспектах существования, разработки, эксплуатации и модернизации ИС, которые наиболее подвержены изменениям. Неудивительно, что подавляющее большинство специалистов в области компьютерных наук определяют ИС прежде всего как программные системы. Между тем, такое восприятие ИС неоправданно узкое и может приводить к значительным ошибкам, вызванным неверной оценкой назначения, важности и способов эксплуатации остальных компонентов ИС.

Для того чтобы сформулировать определение термина «информационная система», рассмотрим вначале официальные стандарты в данной области. К этим стандартам относится, прежде всего, комплекс межгосударственных общих стандартов ГОСТ 34 «Информационная технология». Данный комплекс был разработан в конце 1980-х – начале 1990-х гг. и до сих пор является главным стандартом в области ИС и информационных технологий (ИТ) для стран СНГ. В соответствии с положениями ГОСТ 34.003-90 «Автоматизированные системы. Термины и определения», термин «Автоматизированная система» определяется как система, состоящая из персонала и комплекса средств автоматизации его деятельности, реализующая информационную технологию выполнения установленных функций. Под термином «информационная технология» в этом стандарте понимаются приемы, способы и методы применения средств вычислительной техники при выполнении функций сбора, хранения, обработки, передачи и использования данных. Термином «функция автоматизированной системы» обозначается совокупность действий автоматизированной системы, направленная на достижение определенной цели [1].

Согласно действующему в Украине стандарту ДСТУ 2226-93 «Автоматизовані системи. Терміни і визначення», под термином «автоматизированная система» понимается организационно-техническая система, состоящая из средств автоматизации определенного вида или нескольких видов деятельности

людей и персонала, выполняющего эту деятельность. В то же время в стандарте ДСТУ 2941-93 «Розроблення систем. Терміни та визначення» термин «автоматизированная система» трактуется как система, реализующая информационную технологию выполнения установленных функций при помощи персонала и комплекса средств автоматизации, что в целом повторяет определения стандарта ГОСТ 34.003-90.

Таким образом, основным термином действующих стандартов является термин «автоматизированная система». Учитывая, что в любую автоматизированную систему входят ИТ, реализация которых позволяет достичь поставленные перед системой цели, можно считать любую автоматизированную систему также информационной системой.

В то же время за 1990-е – 2000-е гг. сложился целый ряд определений термина «информационная система» как самостоятельно существующего термина. Так, в [2] ИС характеризуется как система, направленная на хранение, выбор и модификацию постоянно существующей информации. В [3] под ИС понимается система, способная генерировать новую информацию, сохранять сгенерированную информацию и воспринимать (рецептировать) сгенерированную или хранимую информацию. При этом дополнительно к указанным свойствам ИС должна использовать информацию для достижения цели, а также извлекать в процессе обработки имеющейся информации ценную информацию. В [4] ИС трактуется как комплекс, включающий вычислительное и коммуникационное оборудование, программное обеспечение, лингвистические средства и информационные ресурсы, а также системный персонал и обеспечивающий поддержку динамической информационной модели некоторой части реального мира для удовлетворения информационных потребностей пользователей.

Эти и другие определения показывают важность организации процессов хранения информации и обработки хранимой информации для разработки эффективных и качественных ИС.

Следует отметить, что аналогичное мнение господствует и за рубежом. Так, например, методология структурного анализа и проектирования ИС (Structured System Analysis and Design Method, SSADM), принятая Великобританией в качестве государственного стандарта, представляет завершение разработки ИС как создание и, при необходимости, оптимизацию ее физических моделей баз данных. Проблемы разработки программного и технического обеспечений, реализующих ИТ сбора, обработки и отображения данных, выходят за пределы данной методологии [5].

Эти и им подобные определения позволяют рассматривать ***ИС*** как ***системы, состоящие из персонала и комплекса средств автоматизации и направленные на достижение главной цели деятельности ИС – формирование и отображение единого целостного информационного представления объекта или процесса в соответствии с поставленными перед системой целями***. Под ***комплексом средств автоматизации*** следует понимать ***вычислительное и коммуникационное оборудование, программное обеспечение,***

лингвистические средства и информационные ресурсы, нормативные и распорядительные документы и прочие средства, обеспечивающие главную цель деятельности ИС. Термин ***«информационная технология»*** здесь и в дальнейшем следует понимать как ***совокупность унифицированных методов, приемов и способов применения комплекса средств автоматизации или его отдельных элементов, используемую для выполнения одной или нескольких функций ИС***. Под ***функцией ИС*** следует понимать ***совокупность операций ИС, выполняемых над данными и направленных на достижение поставленной перед ИС конкретной цели или подцели***.

Основываясь на этих определениях, далее рассмотрим особенности ИС, возникшие в процессе развития подобных систем.

1.2 Развитие подходов к разработке функциональной структуры информационных систем

Как уже было отмечено выше, основной целью деятельности ИС является формирование и отображение единого целостного информационного представления объекта или процесса в соответствии с поставленными перед системой целями. Тогда главным фактором, определяющим развитие ИС, следует признать создание и совершенствование подходов к организации совокупности функций ИС, реализация которых должна привести к достижению цели деятельности ИС. Структуру такой совокупности функций ИС с учетом их взаимосвязей будем в дальнейшем называть функциональной структурой (ФС).

До конца 1960-х – начала 1970-х гг. разработка ИС в их современном представлении являлась весьма сложным процессом, а сами ИС были уникальными продуктами. Поэтому выработка общих подходов к разработке ФС ИС началась, фактически, только в конце 1960-х гг. К этому времени накопленный опыт создания и эксплуатации ИС позволил выявить наиболее общие закономерности создания ФС ИС [6-9].

На основе этих закономерностей стало возможным проведение в конце 1960-х – начале 1970-х гг. работ по выделению функций ИС, которые являются типовыми, повторяющимися с небольшими отклонениями на многих объектах или процессах схожего назначения. Результатами этих работ стали совокупности типовых задач, решение которых позволяло реализовать соответствующую типовую функцию ИС [10-14]. Такие задачи стали называть функциональными задачами (ФЗ). Тогда же было предложено группировать отдельные ФЗ в функциональные подсистемы – совокупности ФЗ, объединенных для управления каким-либо подразделением, ресурсом или же процессом. Примерами функциональных подсистем, предназначавшихся для управления ресурсами предприятия, являются подсистемы «Управление кадрами», «Управление финансами». Примерами функциональных подсистем, предназначавшихся для

управления отдельными процессами предприятия, являются подсистемы «Технико-экономическое планирование», «Технологическая подготовка производства», «Материально-техническое снабжение», «Оперативное управление основным производством», «Распределение и сбыт продукции» [14].

Необходимо отметить, что в это же время аналогичные изыскания происходили и в других странах. Так, например, первая ИС R/1 одного из современных лидеров в области разработки и сопровождения ИС – компании SAP AG – появилась в 1970-х гг. В основе этой системы лежало решение по автоматизации финансового учета, разработанное в 1973 г. [15].

Выделение типовых ФЗ ИС и сведение их в функциональные подсистемы было обусловлено, прежде всего, особенностями организации объектов и процессов, для которых разрабатывались ИС. В 1970-х гг. управление большинством предприятий осуществлялось на основе бюрократических и функциональных организационных структур [16]. Подразделения и службы предприятия создавались для выполнения определенных функций, причем функции одного подразделения практически не дублировались функциями другого. Это, в свою очередь, требовало объединить ФЗ ИС в функциональные подсистемы по признаку общности автоматизируемых функций. Следует отметить, что в конце 1980-х гг. в литературе, посвященной проблемам проектирования автоматизированных систем управления, утверждалось [17]:

«Развитие АСУ пошло по пути создания функциональных подсистем, аналогично функциональным подразделениям административно-организационного управления. Этим определяется и структура системы, и состав решаемых в подсистемах задач».

Однако там же также признавалось следующее:

«В перспективе можно ожидать перехода к принципиально иному подходу, когда в основу построения подсистем будет положена структура технологического процесса, для которого создается система управления.

По мере движения материального потока будут последовательно решаться задачи управления его движением, необходимой технологической обработки, обеспечения энергией, транспортными операциями и т.д.».

В это же время выделение функциональных подсистем привело к формированию представления ИС как сложной системы, разработка и/или внедрение которой в полном объеме требует чрезвычайно высоких затрат финансов и времени. Попыткой выхода из этой ситуации стало представление функциональных подсистем ИС как самостоятельных ИС, способных интегрироваться в систему более высокого уровня. Однако в 1970-1980-е гг. это представление не рассматривалось как основа процессов проектирования ИС.

Кроме того, опыт разработки и внедрения ИС выявил еще одну проблему – взаимосвязи ИС и документов, циркулирующих на управляемом объекте. Сущность этой проблемы состоит в том, что в ИС данные из документов, как правило, попадают тогда, когда эти документы осуществят свой жизненный цикл (то есть будут созданы, обсуждены, проверены, согласованы, утверждены и т.п.).

Поэтому принято говорить, что документы в ИС не ведутся, а проводятся. Автоматизацию операций по ведению документов в рамках их жизненного цикла с 1970-х гг. стали возлагать на специализированные ИС – системы электронного документооборота.

В 1980-х гг. происходило развитие подходов к формированию иерархических ФС ИС для различных отраслей хозяйствования и типов предприятий [17–21]. Однако развитие идей, концепций и стандартов логистического управления, а также новых типов организаций, структура которых относится к структурам дивизионального типа [16], привело во второй половине 1980-х гг. к переосмыслению подходов к разработке ФС [22, 23]. В это время был также сформулирован термин «контур управления», который описывал результат декомпозиции всей совокупности автоматизируемых функций управления предприятием на отдельные подсистемы по признаку управления состоянием отдельных ресурсов или типов ресурсов [24].

Применение идей ресурсного управления к разработке систем управления объектами и процессами (в том числе ИС) привело к появлению наборов типовых функций ИС, реализация которых могла бы обеспечить повышение эффективности и/или качества управления объектом или процессом. Рассмотрим особенности таких наборов на примере функций стандарта Manufacturing Resourse Planning II (MRP II), разработанного в США в 1980-х гг. Этот стандарт часто характеризуется как один из наиболее распространенных методов управления производством и дистрибуции в мире [25][1].

В своем развитии стандарт MRP к концу 1980-х гг. прошел следующие этапы:

а) 1960-1970-е гг. – планирование потребностей в материалах, на основании данных о запасах на складе и состава изделий (Material Requierment Planning);

б) 1970-1980-е гг. – планирование потребностей в материалах по замкнутому циклу (Closed Loop Material Requirment Planning), включающее составление производственной программы и ее контроль на цеховом уровне;

в) конец 1980-1990-е гг. – ведение прогнозирования, планирования и контроля за производством на основе данных, полученных от поставщиков и потребителей.

Функции управления производственным предприятием в соответствии с MRP II в конце 1980-х гг. были разделены на шестнадцать следующих групп [25]:

а) «Sales and Operation Planning» («Планирование продаж и производства»);

б) «Demand Management» («Управление спросом»);

в) «Master Production Scheduling» («Составление плана производства»);

[1] Как видно из предыдущего текста, примерно до конца 1980-х гг. развитие подходов к организации ФС ИС в СССР, США и в странах нынешнего Евросоюза шли в схожих направлениях и примерно на одинаковом уровне. Однако распад СССР и хаос, наступивший в 1990-х гг. в странах СНГ, серьезно затормозил развитие научных подходов к организации и созданию ИС. Особенно характерно это для Украины, в которой доля ИС и ИТ отечественной разработки на рынке крайне мала.

г) «Material Requirement Planning» («Планирование материальных потребностей»);

д) «Bill of Materials» («Спецификации продуктов»);

е) «Inventory Transaction Subsystem» («Управление складом»);

ж) «Scheduled Receipts Subsystem» («Плановые поставки»);

и) «Shop Flow Control» («Управление на уровне производственного цеха»);

к) «Capacity Requirement Planning» («Планирование производственных мощностей»);

л) «Input/output control» («Контроль входа/выхода»);

м) «Purchasing» («Материально-техническое снабжение»);

н) «Distribution Resourse Planning» («Планирование распределения ресурсов»);

о) «Tooling Planning and Control» («Планирование и контроль производственных операций»);

п) «Financial Planning» («Управление финансами»);

р) «Simulation» («Моделирование»);

с) «Performance Measurement» («Оценка результатов деятельности»).

Рассмотренный перечень групп функций является наглядным доказательством справедливости приведенного выше утверждения о переходе к процессному представлению систем управления (в том числе ИС). Среди групп функций данного перечня можно выделить не только традиционные группы административно-организационного управления (пункты е), и), п)), но и группы, ориентированные на процессное управление. При этом целью ИС, построенных на основе стандарта MRP II, является интеграция всех основных процессов, реализуемых предприятием (снабжение, запасы, производство, продажа, планирование, контроль за выполнением плана, затраты, финансы, основные средства и т.д.). Назначением таких ИС является оптимальное формирование потоков материалов (сырья), полуфабрикатов (в том числе находящихся в производстве) и готовых изделий.

В начале 1990-х гг. развитие идей типизации проектных решений, концепций ресурсного и процессного управления, а также распространение проектных форм организационных структур заставило пересмотреть способы представления ФС ИС. Возникла проблема максимально эффективного использования типовых проектных решений (ТПР) для синтеза ФС ИС, предназначенных для управления уникальными предприятиями или же предприятиями, особенности которых в значительной степени влияют на осуществление типовых бизнес-процессов (БП). Кроме того, быстрое изменение структур организаций в соответствии с изменением их целей деятельности привело к необходимости решения проблемы открытости ФС ИС [26] для изменений и добавлений отдельных элементов и связей.

Результатом решения данных проблем стало представление ФС ИС как совокупности отдельных функциональных модулей (ФМ) [27–31]. ФМ выделяются из всей совокупности автоматизируемых функций по признакам общности управляемых ресурсов и общности операций, которые осуществляют при

управлении данными ресурсами в управляемом процессе. При этом ФМ в значительной степени независимы друг от друга, а интеграция отдельных ФМ в ФС ИС производится обычно либо путем создания специализированного базового ФМ [29, 30], либо путем создания специализированных ИТ информационного взаимообмена между задачами отдельных ФМ. Такой подход к разработке ФС ИС потребовал жесткой стандартизации представления ФС ИС на макроуровне (процедуры обмена данными между отдельными ФМ, схемы подключения отдельных ФМ, процессы координации решения ФЗ различных ФМ и т.д.) и на микроуровне (внутренняя структура ФМ, процедуры обмена информацией между отдельными задачами одного ФМ, схемы настроек ФМ на особенности объекта или процесса и т.д.).

Характерной чертой ФМ является снижение влияния на их структуру как особенностей подразделений, так и особенностей процессов управляемого предприятия. Предполагается, что задачи, образующие ФМ ИС, являются подмножеством ФЗ, наиболее предпочтительным для управления большинством предприятий и организаций одного и того же типа. Кроме того, независимость ФМ друг от друга, а также жесткая стандартизация обмена данными и командами между ФМ, окончательно сформировали представление каждого отдельного ФМ как самостоятельной ИС.

Следует отметить, что в начале 1990-х гг. возникло новое направление исследований – реинжиниринг, которое занимается изучением проблем повышения эффективности деятельности предприятия за счет кардинального изменения его БП [32–37]. При этом в качестве одного из способов реинжиниринга рассматривается внедрение и эксплуатация на предприятии современных ИТ, в том числе и ИС [33, 35]. Однако в 1990-х гг. специалисты по реинжинирингу предприятий уделяли недостаточно внимания проблемам выявления и описания взаимосвязей между изменяемыми БП, организационной структурой предприятия и ФС внедряемой ИС, хотя факты наличия подобных взаимосвязей ими признавались [37].

Рассмотренные особенности представления ИС в 1990-х гг. привели к формированию компанией Gartner концепции Enterprise Resource Planning (ERP, планирования ресурсов предприятия). Данная концепция представляет ИС как результат интеграции отдельных (в том числе, разнородных) ФМ в единую целостную систему управления процессами объекта. Ключевая особенность ИС, основанной на концепции ERP (ERP-ИС), состоит в интеграции данных из всех аспектов деятельности управляемого объекта. Для этого используется единая база данных и многочисленные программные модули, обеспечивающие выполнение различных функций ИС [38].

В основе ERP-ИС лежит развитие идеи планирования и управления ресурсами предприятия, рассмотренной в стандартах серии MRP. Однако в отличие от стандарта MRP II, в ERP-ИС основной является концепция процессного управления объектом. Согласно этой концепции, управляемый объект рассматривается как совокупность (в частном случае – последовательность) процессов,

выполнение которых обеспечивает достижение поставленных перед объектом целей (получение прибыли, снижение затрат и т.п.). Поэтому ФМ, образующие ERP-ИС, рассматриваются как ИС, ориентированные на информатизацию отдельных процессов и способные функционировать отдельно друг от друга.

Развитие концепции ERP привело к появлению в начале 2000-х гг. концепции ERP II. Эта концепция рассматривает не только организацию планирования и управления ресурсами отдельного предприятия, но и организацию совместной деятельности целого ряда различных предприятий, объединенных общими интересами. Основой для такого объединения становятся решения в области управления взаимоотношениями с клиентами, управления цепочками поставок, электронной коммерции и взаимодействия ФМ «front office» и ФМ «back office»[2] [39].

В настоящее время большинство ИС, основанных на концепциях ERP / ERP II, включают в себя ФМ, которые реализуют следующую функциональность [38, 39]:

а) Customer Relationship Management Systems (CRM) – системы управления взаимодействием с клиентом, которые используются для повышения уровня продаж, оптимизации маркетинга и улучшения обслуживания клиентов путём выполнения следующих действий: сохранение информации о клиентах и истории взаимоотношений с ними; установление и улучшение бизнес-процедур; последующий анализ результатов;

б) Enterprise Asset Management Systems (EAM) – системы управления основными фондами предприятия, которые используются для сокращения затрат на техобслуживание, ремонт и материально-техническое снабжение без снижения уровня надежности;

в) Enterprise Content Management Systems (ECM) – системы управления информационными ресурсами объекта (управления корпоративной информацией), которые используются для поддержки единого жизненного цикла неструктурированной информации (контента) различных типов и форматов (иногда ECM трактуются как системы электронного документооборота, что может быть не совсем верно);

г) Human Resource Management Systems (HRM) – системы управления персоналом, которые используются для обеспечения организации персоналом, способным выполнять возложенные на него трудовые функции, и оптимального использования этого персонала;

д) Manufacturing Execution Systems (MES) – производственные исполнительные системы, которые решают задачи синхронизации, координируют, анализируют и оптимизируют выпуск продукции в рамках какого-либо производства;

е) Supply Chain Management Systems (SCM) – системы управления цепочками поставок (управления запасами), которые используются для управления

[2] Термин «front office» используется для обозначения функциональности, ориентированной на внешних (по отношению к управляемому объекту) пользователей (заказчиков, клиентов, представителей других предприятий и т.п.). Термин «back office» используется для обозначения функциональности, ориентированной на сотрудников управляемого объекта и не предназначенной для «широкой публики».

всеми этапами снабжения управляемого объекта и контроля движения товаров на этом объекте;

ж) Warehouse Management Systems (WMS) – системы управления складами, которые обеспечивают автоматизацию и оптимизацию всех процессов складской работы профильного предприятия.

Данный перечень можно продолжать в зависимости от особенностей управляемого объекта, его процессов и тех концепций и моделей, которые используются для информатизации управления этими процессами. Так, для информатизации управления техобслуживанием и ремонтом в рамках ERP-ИС могут применяться как EAM-системы, так и CMMS-системы (Computerized Maintenance Management Systems, компьютеризированные системы управления техническим обслуживанием), которые могут включаться в EAM-системы.

Примерно в это же время (конец 1990-х – начало 2000-х гг.) отмечается обратное влияние: предприятие проводит изменение своей организационной структуры и БП, основываясь на особенностях эксплуатации ИС, ФС которой построена по модульному принципу. Направление изменений в этом случае определяется особенностями автоматизируемого БП.

Влияние автоматизируемых БП на ФС проектируемой ИС до недавнего времени осуществлялось, главным образом, за счет применения в ИС конкретных бизнес-правил и формализующих эти правила алгоритмов решения ФЗ [24, 40–32]. Однако, начиная с 1990-х гг., влияние БП на ИС в целом и на ФС ИС в частности сказывается, главным образом, на переходе от структурных методов организации и автоматизации управления хозяйственной деятельностью к процессным методам [27–31, 43]. Данный переход также способствовал представлению ФС ИС как совокупности отдельных, слабо зависящих друг от друга ФМ.

Однако в ходе развития концепций ERP и ERP II выявился целый ряд проблем [38], порожденных разработкой, внедрением и эксплуатацией ERP-ИС. Эти проблемы можно разделить на такие группы:

а) проблемы, вызванные взаимовлиянием ERP-ИС и БП управляемого объекта, к которым относятся:

- сложности согласования предлагаемых изменений БП управляемого объекта с оптимальным использованием ERP-системы, эксплуатируемой на этом объекте;
- возможность потери конкурентоспособности управляемого объекта вследствие перепроектирования его БП под «промышленный стандарт», поддерживаемый ERP-ИС;
- наличие в ERP-ИС избыточных ФЗ (по сравнению с фактическими потребностями управляемого объекта);

б) проблемы, связанные с затратами, к которым относятся:

- высокая стоимость покупки, внедрения и сопровождения ERP-ИС;
- высокие затраты на переход к другой версии ERP-ИС или же на переход к ERP-ИС другого производителя (что снижает гибкость и стратегический контроль на корпоративном уровне);

в) проблемы, связанные с персоналом управляемого объекта, к которым относятся:

- зависимость успеха внедрения ERP-ИС от квалификации и опыта персонала, в том числе от результатов обучения способам обеспечения безошибочной работы ИС;
- возможность снижения эффективности работы управляемого объекта в целом из-за неверных действий сравнительно небольшой группы пользователей ERP-ИС (проблема «слабого звена») или из-за неверных данных, вводимых отдельными пользователями;

г) проблемы создания и модернизации ERP-ИС, к которым относятся:

- низкая гибкость ERP-ИС и трудности их адаптации к потокам данных и БП конкретных объектов;
- ограничение возможности индивидуальной доработки ERP-ИС;
- проблемы совместимости с устаревшими ИС предприятий-партнеров;

д) проблемы, связанные с эксплуатацией ERP-ИС, к которым относятся:

- сложности эксплуатации ERP-ИС;
- возможные проблемы с отчетностью, сферами ответственности и моральным состоянием сотрудников управляемого объекта из-за «стирания границ ответственности», вызванного внедрением и эксплуатацией ERP-ИС на управляемом объекте;
- конфликт концепций ERP и ERP II с мерами по нераспространению секретной информации между подразделениями управляемого объекта.

Примерно в это же время проявился кризис, возникший в ходе организации эксплуатации ИС и ИТ на конкретных предприятиях и организациях. С одной стороны, ИТ-специалисты, увлеченные решением технических вопросов разработки, внедрения и сопровождения ИС и ИТ, не понимали (и в ряде случаев искренне), почему остальные сотрудники предприятия не горят желанием пользоваться «технически красивыми» решениями, которые они предлагают. Сосредоточение деятельности ИТ-специалистов на решении вопросов чисто технического управления ИС и ИТ заслоняло от сотрудников предприятий и организации важность использования этих ИС и ИТ для бизнес-деятельности предприятия. С другой стороны, сотрудники и руководство предприятий и организаций попросту не понимают, чем именно заняты ИТ-специалисты, которым поручалось решать вопросы эксплуатации ИС и ИТ. При этом руководство предприятий и организаций волнует видимость (а зачастую и фактическое проявление) низкой экономической эффективности деятельности такой организации. Как отмечено в [44]: «Вопреки ожиданиям бизнес получал от ИТ довольно скудную отдачу, при том что расходы на ИТ-отделы оставались всегда достаточно высокими, а возможности ИТ практически не находили полезного применения».

Преодоление этого кризиса стало возможным, прежде всего, благодаря пониманию того факта, что чисто технического управления ИС и ИТ, внедряемы-

ми и эксплуатируемыми на предприятиях и организациях, уже недостаточно. Как отмечается в [45], первым шагом для преодоления подобного кризиса является необходимость «избавиться от комплекса миссионерства и четко осознать, что ИТ-подразделение компании – это сервисная служба, которая оказывает услуги другим функциональным подразделениям, и в этом плане, с точки зрения топ-менеджера, она немногим отличается, например, от службы, осуществляющей уборку офиса». Затем в ходе преодоления этого кризиса подразделение, которому поручается решать вопросы эксплуатации ИС и ИТ на предприятии, реорганизуется в обычное (с виду) подразделение предприятия или организации. Это подразделение занимается выполнением вспомогательного БП, обеспечивающего повышение эффективности и качества выполнения основных БП предприятия. К руководству такой организацией приходят люди, сочетающие понимание технических и экономических задач управления ИС и ИТ.

В ходе преодоления данного кризиса в начале 2000-х гг. были сформированы основные принципы и выделены основные практики экономического и технико-экономического управления ИС и ИТ предприятий и организаций. Эти практики постепенно формализуются и согласовываются со всеми заинтересованными сторонами (руководством предприятия, руководством и сотрудниками отдельных подразделений предприятия, которые используют в своей деятельности ИС и ИТ).

Попытки устранения недостатков «классических» ERP-ИС и им подобных систем и технологий, а также результаты преодоления кризиса в управлении эксплуатацией ИС и ИТ привели в первой половине 2000-х гг. к четкому осознанию необходимости адаптации типовых ФЗ ИС и реализующих эти задачи ИТ под уникальные особенности процессов конкретных предприятий и организаций. Кроме того, оказалось, что целый ряд ФЗ, входящих в типовые функциональные подсистемы или ФМ ИС, нецелесообразно использовать для выполнения процессов конкретных предприятий/организаций или управления этими процессами. В результате решения этих проблем возник подход к формированию представления ФС ИС как совокупности взаимосвязанных экономически целесообразных ИТ-услуг[3], своевременное предоставление и выполнение которых обеспечивает эффективную и качественную деятельность управляемого объекта и/или процесса (сервисный подход). Под процессом здесь и в дальнейшем следует понимать совокупность взаимосвязанных и взаимодействующих видов деятельности, преобразующих входы (материальные, информационные потоки и т.п.) в выходы, представляющие ценность для потребителя (согласно стандарту ISO 9000:2005).

Сервисный подход приобретает особое значение в современных условиях дефицита финансовых ресурсов и высоких рисков создания и внедрения ИС и ИТ на современных предприятиях. Подобные условия определяют главные

[3] Термин «ИТ-услуга» предлагается для устранения многозначности толкования оригинального термина «IT-service», который может использоваться для описания самых различных элементов ИС. Детальное определение термина «ИТ-услуга» будет дано далее.

преимущества сервисного подхода к формированию ФС ИС как возможности ускорения формирования новых вариантов ФС ИС, увеличения производительности разработки ИС и повышения гибкости реакции элементов комплекса средств автоматизации на изменение БП [46]. Однако на практике руководство предприятия склонно забывать о необходимости эффективного управления данными и отдельными ИТ-услугами [47]. В результате этого затраты финасовых и других ресурсов на эксплуатацию отдельных ИТ-услуг ИС предприятия становятся неоправданными и не могут окупиться за счет эффекта от эксплуатации ИС предприятия в целом.

Другой, не менее важной проблемой информатизации предприятий, является уже отмеченное выше разнообразие поставщиков и решений на рынке ИТ-услуг ИС. Такое разнообразие приводит к тому, что ИС целого ряда предприятий формируются из разнородных ИТ-сервисов, обеспечивающих оказание соответствующих ИТ-услуг. Вследствие этого возникают проблемы повышения эффективности использования ИТ в основной деятельности предприятия и к оптимизации затрат расходуемых при этом ресурсов различного рода [48]. Эту задачу не следует считать элементарной или же типовой – существует большое количество примеров того, как работы по информатизации предприятия не дают желаемого эффекта или же приводят к излишним трудозатратам. Неудачные решения этой задачи породили эффект, который в [49] назван «ИТ-слепотой» (IT blindness) – неспособностью существующих ИС и ИТ «увидеть» и оценить реальные процессы в той среде, в которую они включены. Поэтому проблему формирования такой ФС ИС, которая, с одной стороны, была бы наиболее предпочтительной для представителей объекта управления, а с другой – соответствовала бы возможностям предприятия-разработчика ИС, в настоящее время следует считать нерешенной.

1.3 Процессно-архитектурное представление жизненного цикла информационной системы

Одним из подходов к решению рассмотренной выше проблемы формирования варианта конфигурации ФС ИС, который, с одной стороны, был бы наиболее предпочтительным для представителей объекта управления, а с другой – соответствовал бы возможностям предприятия-разработчика ИС, является формальное осмысление способов интеграции отдельных элементов ИС и специализированных ИТ в единую непротиворечивую систему управления предприятием. Иными словами, необходимо на концептуальном и формальном уровнях выделить законы, закономерности, модели и методы построения современных ИС из большого количества разнородных элементов.

В настоящее время основой для проведения подобных исследований следует считать сложившийся в мире к 2000-м гг. подход, обобщающий множество

разнообразных концепций, теоретических разработок и прикладных работ в рамках системы стандартов. Подобные стандарты представляют собой совокупность наиболее удачных практик, применение которых для решения прикладных задач в подавляющем большинстве случаев гарантирует успешные результаты.

Один из подобных стандартов, а именно стандарт ISO/IEC 15288 «System Engineering – System life cycle processes» [50] устанавливает современную практику описания жизненного цикла системы на основе возможного использования абстрактной функциональной модели, представляющей концептуализацию потребности в системе, ее реализации, применения, развития и ликвидации. При построении такой модели предполагается, что система развивается на протяжении жизненного цикла в результате действий, осуществляемых и управляемых людьми, работающими в организациях и использующими определенные процессы в своей деятельности. Детали модели жизненного цикла выражаются в терминах этих процессов, названных процессами жизненного цикла, при помощи которых может быть смоделирован жизненный цикл системы, их результатов, взаимосвязи и возникновения. Функции, которые осуществляются данными процессами, определяются в зависимости от конкретных целей, результатов и набора действий, составляющих данный процесс.

Процессы жизненного цикла системы в общем случае обладают такими особенностями [50]:

- могут применяться любой организацией при приобретении и использовании или создании и поставке системы;
- распространяются на любой уровень системной иерархии и на любую стадию жизненного цикла;
- основываются на принципах модульности (максимальная слаженность функций процесса и минимальная связь между процессами) и собственности (процесс связывается с ответственностью);
- не препятствуют и не исключают использование дополнительных процессов, которые организация посчитает полезными.

Тогда отдельный жизненный цикл системы, согласно [50], может быть представлен как сложная система процессов, обычно обладающих параллельными, итеративными, рекурсивными и зависящими от времени характеристиками.

Процессы могут выполняться параллельно в рамках проекта (например, проектные действия и действия по подготовке к созданию системы выполняются одновременно) или между проектами (например, когда системные элементы разрабатываются одновременно в различных проектах).

Итеративное использование процессов, то есть повторное применение процесса или множества процессов на одном уровне иерархии, имеет важное значение для постоянного уточнения результатов процесса, например взаимодействие между последовательными действиями по верификации и действиями

по комплексированию может постепенно укреплять уверенность в соответствии продукта предъявленным требованиям.

Рекурсивное использование процессов, то есть повторное применение одного и того же процесса или множества процессов к последовательным уровням детализации иерархической структуры системы, является ключевым аспектом применения настоящего стандарта. Результаты процессов на любом уровне, будь то информация, артефакты или услуги, являются входами для таких же процессов, но реализуемых на более низком или более высоком уровнях. В итоге возникает ответная информация, артефакты или услуги, которые могут модифицировать первоначальный выход процесса. Таким образом, результаты процессов, полученные на всех уровнях системной архитектуры, могут быть согласованы и совместимость их достигнута, например, в виде описаний системных элементов, формирующих системную архитектуру.

Изменяющийся характер воздействий на систему (например, изменения среды функционирования, новые возможности реализации системных элементов, модифицированная структура и обязанности в организациях) требует постоянной проверки выбора и синхронизации использования процессов. Таким образом, применение процесса в течение жизненного цикла является интенсивно меняющимся во времени действием, реагирующим на множество внешних воздействий на систему.

Процессы жизненного цикла системы в общем случае делятся на четыре группы процессов [50]:

- соглашения;
- предприятия;
- проекта;
- технические процессы.

Группа процессов соглашения определяет действия, необходимые для достижения соглашения между двумя организациями и состоит из двух процессов – процесса приобретения и процесса поставки.

Группа процессов предприятия управляет способностью организации приобретать и поставлять продукцию или услуги посредством запуска проектов, их поддержки и контроля. Процессы предприятия обеспечивают ресурсы и инфраструктуру, необходимые для осуществления проектов, и гарантируют достижение целей и исполнение обязательств организации по соглашениям. Эти процессы не рассматриваются в качестве исчерпывающей совокупности бизнес-процессов, которые делают возможным стратегическое управление деятельностью организации.

Группа процессов проекта используется для установления и выполнения планов, оценки фактических достижений и продвижений проекта в соответствии с планами и для контроля выполнения проекта вплоть до его завершения. Отдельные процессы проекта могут осуществляться в любой момент жизненного цикла и на любом уровне иерархии проектов как в соответствии с проектными планами, так и с учетом непредвиденных обстоятельств. Уровень точности и

формализации, с которыми осуществляются процессы проекта, зависит от сложности самого проекта и проектных рисков.

Группа технических процессов используется для определения требований к системе, преобразования этих требований в эффективный продукт, позволяющий осуществлять, при необходимости, устойчивое воспроизводство этого продукта, использовать его для обеспечения требуемых услуг, поддерживать обеспечение этими услугами и удалять продукт, когда он изымается из обращения.

Отмеченное выше позволяет указать на группу технических процессов жизненного цикла системы как на основной фактор, определяющий общий подход к описанию и моделированию системы (и, в частности, ИС). Поэтому следует уделить особое внимание рассмотрению процессов данной группы.

Группа технических процессов включает в себя процессы [50]:

а) определения требований правообладателей;

б) анализа требований;

в) проектирования архитектуры;

г) реализации элементов системы;

д) комплексирования;

е) верификации;

ж) передачи;

и) валидации;

к) функционирования;

л) технического обслуживания;

м) изъятия и списания.

Анализ описаний технических процессов, а также особенностей процессов других групп показывает, что ключевым понятием, задающим особенности анализа и синтеза систем, моделей их жизненных циклов и других аспектов существования является понятие «системная архитектура» или, точнее, «архитектура системы». В связи с этим целесообразно рассмотреть развитие трактовок данного понятия и прикладные аспекты применения понятия «архитектура системы» в процессах жизненного цикла ИС.

Большой энциклопедический словарь трактует понятие «Архитектура» как «искусство проектировать и строить здания и др. сооружения (также их комплексы), создающие материально организованную среду, необходимую людям для их жизни и деятельности, в соответствии с назначением, современными техническими возможностями и эстетическими воззрениями общества» [51]. В то же время Оксфордский словарь английского языка сообщает о наличии двух значений термина «архитектура» [52]:

а) «искусство или практика проектирования и строительства зданий (школы архитектуры или дизайна); стиль, в котором здание спроектировано и построено, особенно по отношению к определенному периоду времени, места или культуры (грузинская архитектура)»;

б) «комплекс или тщательно разработанная структура чего-либо (химическая архитектура человеческого мозга); концептуальная структура и логическая организация компьютера или компьютерной системы».

Большинство определений понятия «архитектура» в области компьютерных наук фокусируется на артефактах, используемых для описания архитектуры во втором значении этого термина [53]. Однако и в этом случае описание архитектуры рассматривается как набор практик в рамках искусства или как практики архитектуры (причем такой набор иногда также называют архитектурой). Кроме того, существует единство семантики термина «архитектура», берущей свое начало из строительной отрасли и существующей в настоящее время в гражданской и в других отраслях. В результате определение семантики понятия «архитектура системы» становится весьма затруднительным и запутанным. Как утверждал Захман в работе [54]: «Архитектура есть Архитектура есть Архитектура»[4].

Первоначально большинство определений понятий «Архитектура программного обеспечения (ПО)» и «Архитектура системы» были сосредоточены на структурном аспекте архитектуры по аналогии с идеями архитектуры в строительной отрасли. Например, одно из определений трактовало понятие «Архитектура ПО» следующим образом: «набор архитектурных (или, если угодно, проектных) элементов, которые имеют особую форму» [55]. Позже аналогичные определения появились и для веб-решений: «Архитектура – это множество важных решений по организации системы ПО, по выбору составляющих систему структурных элементов и их интерфейсов вместе с их поведением, определяющим взаимодействие этих элементов, состава структурных и поведенческих элементов в более крупных подсистемах; также архитектурный стиль, который определяет эту организацию – системные элементы и их интерфейсы, их взаимодействие и состав» [56]. В 2000-х гг. данная точка зрения на архитектуру нашла воплощение в работах специалистов компании Microsoft «Metropolis» и «Metropolis and SOA Governance», проводящих аналогии между эволюцией ИТ и процессами эволюции городов и промышленности [57, 58].

Одной из первых попыток узаконить максимально общее определение понятия «Архитектура» для компьютерных наук является стандарт IEEE 1471:2000. В нем предлагается следующее определение: «Архитектура: фундаментальная организация системы, воплощенная в ее компонентах, их взаимоотношении друг с другом и с окружающей средой, а также принципы, определяющие проектирование и эволюцию системы». Однако в последующих редакциях этого стандарта данное определение было серьезно изменено. Так, был удален термин «организация», поскольку он уже имел специализированное значение в контексте стандартов ISO. Также был заменен термин «компоненты», потому что он неправильно трактовался как «программные компоненты».

[4] В той же статье Захман дает такое определение архитектуры: «Архитектура – это полный набор пересечений между Абстракциями и Перспективами и представляет собой набор соответствующих описательных представлений для любого объекта, который может быть создан».

Определение стандарта IEEE 1471:2000 было использовано, расширено и трактовалось в измененном виде многими исследователями в самых различных формах. Например:

а) «Архитектура чего-либо – это:

- его фундаментальная организация, воплощенная в виде его компонентов и их связей друг с другом и с окружающей средой;
- принципы, которые управляют его проектированием и эволюцией» [59];

б) «Архитектура предприятия – это непрерывная практика описания основных элементов социотехнической организации, их отношений друг с другом и с окружающей средой, для понимания сложности и управления изменениями» [60].

Различные авторы вкладывают, в общем-то, близкий, но, тем не менее, отличающийся смысл в этот термин[5] или же выделяют в составе архитектуры различное число составляющих компонентов – от двух до семи и более [48, 61–70]. Так, например, в [61] предлагается трактовать понятие «архитектура ИС» как концепцию, определяющую модель, структуру, выполняемые функции и взаимосвязь компонентов ИС. В [70] предлагается рассматривать архитектуру ИС как устойчивую совокупность структурных, функциональных и потребительских характеристик разрабатываемой системы, наличие которых обеспечивает у нее такие свойства (качества):

- заданная производительность системы (system performance);
- надежность функционирования (reliability);
- защищенность / безопасность эксплуатации системы (safety / security);
- возможность ее эффективного сопровождения и развития (maintenance and evolution).

Иногда при определении понятия «архитектура ИС» дополнительно говорят и об определенных эстетических требованиях к создаваемой ИС [62, 69]: хорошо спроектированная ИС должна быть не только производительной и устойчивой, но также удовлетворять набору признаков привлекательности для пользователя (to be presentable for user).

Важным также является отличие понятия «архитектура ИС» от понятия «структура системы»: если структура системы описывает только статические аспекты ее построения, то архитектура ИС определяет также и динамику поведения (behaivor) соответствующей ИС, а также задает интерфейсы ее взаимодействия с окружающей средой (например, с внешними объектами, другими системами и т.п.) [70].

В настоящее время наметился новый виток в развитии определения понятия «Архитектура». Принятый в 2010 г. стандарт ISO/IEC/IEEE 42010 «Systems and Software Engineering – Architecture Description» дает следующее определе-

[5] Можно сказать, что определение такого понятия, как «архитектура ИС», характеризуется неопределенностью, аналогичной неопределенности Гейзенберга: любая формулировка отражает специфику либо используемого автором определения аппарата формального описания ИС и ее компонентов, либо конкретного подхода автора определения к реализации архитектуры в рамках ФС и обеспечивающей части ИС. Выработать единое определение этого понятия, которое учитывало бы ВСЕ основные аспекты создания и интеграции ИС, на данном этапе следует признать невозможным.

ние понятия «архитектура»: ***«Архитектура [системы]: фундаментальные понятия и свойства системы в окружающей ее среде, воплощенные в ее элементах, отношениях, а также в принципах ее проектирования и развития»*** [53]. Данное определение учитывает основные проблемы, возникающие при попытках прикладного использования понятия «Архитектура» стандарта IEEE 1471:2000, и, по замыслу создателей, является максимально общим определением, пригодным для описания архитектур практически любых систем.

Термин «система» используется в соответствии с конвенцией ISO для того, чтобы показать, что определяемый термин «архитектура» относится к предметной области системы. По сути, термин «система» в приведенном выше определении и в пределах стандарта ISO/IEC/IEEE 42010 является основой для многих других понятий, в том числе [53]:

а) для определения термина «система» в соответствии с положениями стандарта ISO 15288 как «системы, которые созданы человеком и состоят из одного или нескольких следующих элементов: технические средства, программные средства, люди, процессы (например, процесс оценки), процедуры (например, инструкции оператора), основные средства и природные ресурсы (например, вода, объекты живой природы, материалы)»[6] [50];

б) для аналогичного определения понятий «программные продукты» и «сервисы» в соответствии с описаниями стандарта ISO 12207;

в) для определения программно-основанных систем, которые в соответствии с описаниями стандарта IEEE 1471:2000 представляют собой «любую систему, в которой ПО оказывает существенное влияние на проектирование, конструирование, внедрение и развитие системы в целом» и могут включать в себя «отдельные приложения, системы в традиционном смысле этого слова, подсистемы, надсистему, линейки продуктов, семейства продуктов, целые предприятия и другие сосредоточия интересов».

Стандарт ISO/IEC/IEEE 42010 не накладывает никаких ограничений на определение понятия «система» для пользователей данного стандарта. В дополнение к рассмотренным выше определениям типов систем, данный стандарт может быть использован для выражения архитектур концептуальных и природных систем.

Разделение понятий «концепции или свойства» было сделано для поддержки двух разных философий архитектуры без отдания предпочтения какой-либо из них. Речь идет о следующих философиях [53]:

а) архитектура как концепция: архитектура – это умозрительное понятие системы;

[6] Далее в стандарте ISO 15288 от 2005 г. понятие система определяется следующим образом: «Комбинация взаимодействующих элементов, организованных для достижения одной или нескольких поставленных целей». К этому определению дается два примечания: «Система может рассматриваться как продукт или как совокупность услуг, которые она обеспечивает» и «На практике интерпретация данного термина зачастую уточняется с помощью ассоциативного существительного, например, система самолета. В некоторых случаях слово «система» может заменяться контекстным синонимом, например, самолет, хотя это может впоследствии затруднять восприятие системных принципов».

б) архитектура как восприятие: архитектура – это восприятие свойств системы.

С точки зрения любой из этих философий архитектура является абстракцией, а не артефактом. Стандарт ISO/IEC/IEEE 42010 использует другой термин – «описание архитектуры» – для обозначения артефактов, которые могут применяться для выражения и документирования архитектуры. К сожалению, широко распространено использование термина «архитектура», при котором абстрактная идея смешивается с артефактами, использующими эту идею, – например, в области архитектур предприятия [54].

Архитектура системы определяет систему в ее окружающей среде: именно среда определяет совокупность воздействий на систему. Одно из часто упоминаемых различий между понятиями «архитектура системы» и «проект системы» заключается в следующем: архитектура внешне сосредоточена на системе в окружающей ее среде, тогда как проект системы внутренне сосредоточен на системе, не выходя за пределы ее границ.

То, что лежит в основе системы, определяется следующими понятиями:

а) «элементы системы»: компоненты, входящие в состав системы;

б) «отношения», которые могут быть внутренними и внешними по отношению к системе;

в) «принципы проектирования и развития системы».

Различные сообщества, использующие понятие «архитектура», по-разному акцентируют внимание на этих понятиях. Так, в архитектуре ПО часто сосредоточиваются на программных компонентах в качестве элементов системы и взаимодействиях этих компонентов в качестве ключевых отношений. Системная архитектура акцентирует внимание на структурах отдельных подсистем и таких отношениях, как выделение этих подсистем. В архитектуре предприятия основное внимание уделяется принципам. Предлагаемое определение служит признанием того, что все перечисленные выше понятия могут сыграть свою роль в архитектуре.

Принципы проектирования и развития системы теоретически могут быть выделены архитектором путем реверс-инжиниринга из существующей системы или даже открыты в природе – в зависимости от исследуемой системы. Хотя в созданных человеком системах архитектура часто воспринимается как интенциональная позиция, такая интенциональность не является частью рассматриваемого определения.

Для описания взаимосвязей основных терминов и понятий системы и ее архитектуры в стандарте ISO/IEC/IEEE 42010 предлагается ряд концептуальных моделей, выполненных в нотации диаграммы классов UML. Контекстная диаграмма классов, отражающая основные взаимосвязи базовых понятий, приведена на рис. 1.1 [53].

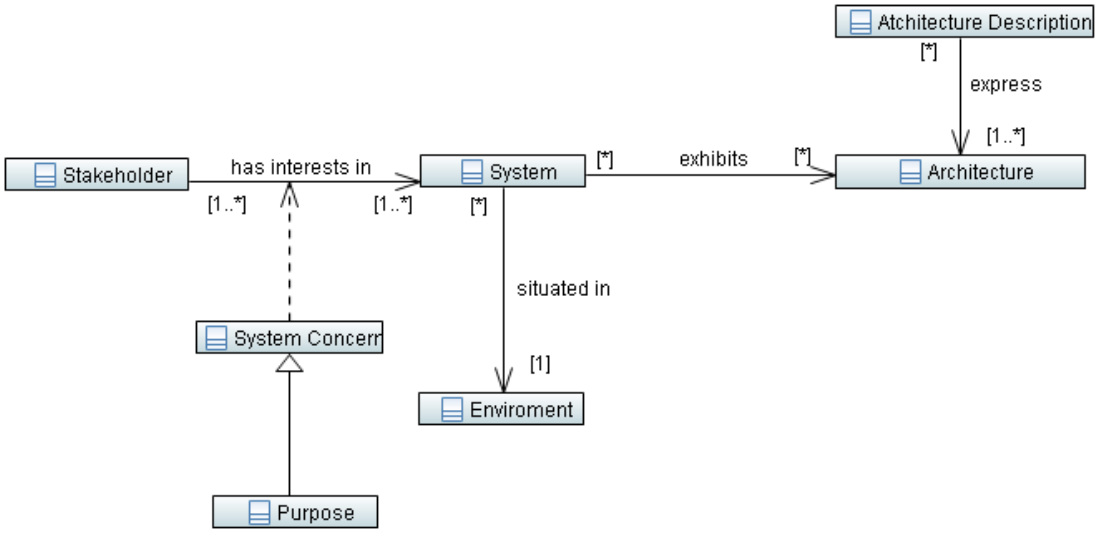

Рисунок 1.1 – Контекстная диаграмма классов понятия «описание архитектуры»

Рассмотрим подробнее основные понятия контекстной диаграммы.

По вопросу «Что именно обозначается термином "система"» данный стандарт не занимает никакой позиции. Пользователи стандарта могут выбирать любую теорию систем из существующих на данный момент. В случае заинтересованности какой-либо системой стандарт предоставляет руководство по документированию архитектуры этой системы.

Каждая система (System) существует в своей окружающей среде (Environment). Система воздействует на эту среду, и наоборот. Окружающая среда системы определяет диапазон воздействий на систему. В стандарте понятие «окружающая среда» представлено в самом широком смысле, включая проектные, операционные, технические, политические, нормативные и другие воздействия на архитектуру системы. Такие воздействия относятся к категории «Проблемы» (System Concern).

Каждая система представляется через архитектуру (Architecture). Описание архитектуры (Architecture Description, AD) представляет собой артефакт, который выражает архитектуры. Архитекторы и другие заинтересованные стороны системы используют AD, чтобы понять, проанализировать и сравнить возможные для системы архитектуры, и часто как своего рода «чертежи» для проектирования и конструирования. AD являются основным предметом стандарта ISO/IEC/IEEE 42010 и находятся в центре внимания на следующей диаграмме (рис. 1.2) [53].

Рассмотрим некоторые понятия и термины данной диаграммы классов подробнее.

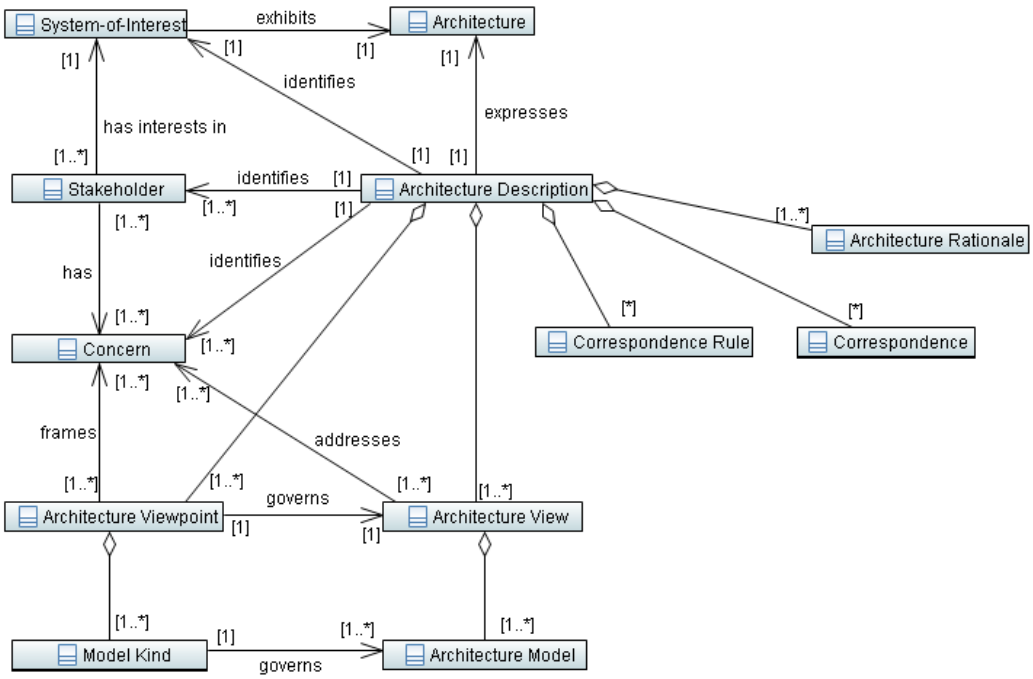

Рисунок 1.2 – Диаграмма классов, описывающая ядро понятия «описание архитектуры»

AD представляет собой рабочий продукт, который используется, чтобы выразить некоторые архитектуры проблемно-ориентированных систем. Стандарт устанавливает требования к AD. AD описывает одну из возможных архитектур проблемно-ориентированной системы и может принимать форму документа, набор моделей, модели хранилища, или какую-либо другую (формат AD стандартом не определен).

Заинтересованная сторона (Stakeholder) – это отдельные лица, группы лиц или организации, имеющие проблемы в проблемно-ориентированных системах. Примеры заинтересованных сторон: клиент, владелец, пользователь, потребитель, проектировщик, сопровождающий, аудитор, специалист по сертификации, архитектор.

Проблема (Concern) – это любой интерес в системе. Примеры проблем: цели (системы), функции, структура, поведение, стоимость, возможности поддержки, безопасность, совместимость.

Архитектурная точка зрения (Architecture Viewpoint) представляет собой набор конвенций конструирования, интерпретации, использования и анализа конкретного типа взгляда на архитектуру. Точка зрения включает в себя виды моделей, точки зрения на языки и нотации, методы моделирования и аналитические методы в рамках определенного набора проблем. Примеры точек зрения: операционные, системные, технические, логические, развертывания, процессные, информационные.

Взгляд на архитектуру (Architecture View) в AD отражает архитектуру проблемно-ориентированной системы с точки зрения одной или нескольких за-

интересованных сторон для решения конкретных проблем, используя конвенции, устанавливающие данную точку зрения.

Взгляд на архитектуру может быть представлен набором архитектурных моделей (Architecture Model). Каждая модель строится в соответствии с конвенциями, учрежденными для данного вида модели, причем эти конвенции, как правило, определяются как часть архитектурной точки зрения. Модели обеспечивают средства для обмена деталями между взглядами на архитектуру и использования множественных обозначений в одном взгляде на архитектуру.

Вид модели (Model Kind) определяет конвенции, задающие правила построения архитектурных моделей.

Для описания связей между элементами, образующими AD, используется понятие «Соответствие». Это понятие, а также понятие «Правило соответствия» используются, чтобы выразить и поддерживать архитектуру отношений, среди которых можно выделить отношения состава, выделения, последовательности, прослеживаемости, зависимости, ограничения и обязательств, действующие в пределах одного AD или между несколькими AD (рис. 1.3) [53].

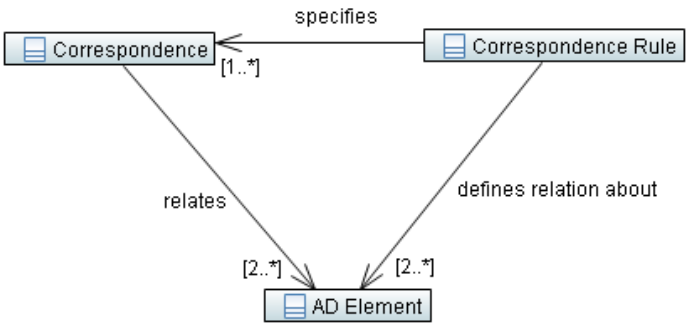

Рисунок 1.3 – Диаграмма классов, описывающая архитектуру отношений

Рассмотрим понятия и термины данной диаграммы классов подробнее.

Любой объект класса AD считается элементом (AD Element). Любые заинтересованные стороны, проблемы, точки зрения, взгляды, виды моделей в AD являются AD-элементами. Когда архитектурная точка зрения или вид модели вводит какие-либо конструкты, они также считаются AD-элементами.

Понятие «соответствие» (Correspondence) выражает отношения между AD-элементами. Соответствия используются для выражения архитектуры отношений в рамках описания архитектуры или между различными архитектурными описаниями. Соответствия могут регулироваться правилами соответствий.

Правило соответствий (Correspondence Rule) поддерживает существование связей в рамках описания архитектуры или между различными архитектурными описаниями.

Создание архитектуры требует формирования архитектурных решений. Стандарт ISO/IEC/IEEE 42010 устанавливает требования к формированию ключевых решений для описания архитектуры в понятиях и терминах, показанных на следующей диаграмме (рис. 1.4) [53].

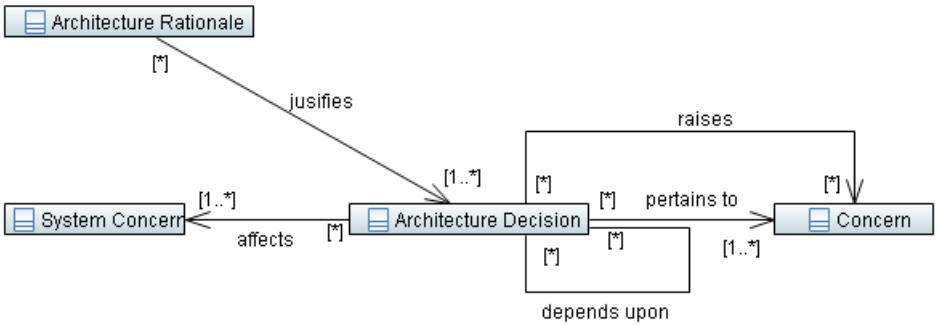

Рисунок 1.4 – Диаграмма классов, описывающая архитектурные решения

Рассмотрим понятия и термины данной диаграммы классов подробнее.

Архитектурное решение (Architecture Decision) влияет на один или несколько AD-элементов и принадлежат одной или нескольким проблемам. Факт принятия архитектурного решения может привести к появлению новых проблем.

Обоснование архитектуры (Architecture Rationale) представляет собой совокупность записей, содержащих объяснения, оправдания или рассуждения о сформированных архитектурных решениях или архитектурных альтернативах, которые не были выбраны.

В архитектуре широко используются два механизма: архитектурный фреймворк и языки описания архитектуры. Концептуальные модели каждого из этих механизмов можно сформировать, основываясь на рассмотренных выше концептуальных моделях архитектуры и описания архитектуры.

Стандарт ISO/IEC/IEEE 42010 «Systems and Software Engineering – Architecture Description» дает следующее определение понятия «архитектурный фреймворк»: *«**Архитектурный фреймворк: конвенции, принципы и методы описания архитектуры, установленные в конкретной области применения и/или сообществом заинтересованных сторон**»* [53].

Архитектурный фреймворк соответствует международным стандартам IS, если он обозначает:

- информацию, идентифицирующую фреймворк;
- одну или несколько проблем;
- одну или несколько заинтересованных сторон, у которых есть указанные выше проблемы;
- одну или несколько архитектурных точек зрения (и их технические спецификации, соответствующие IS);
- правила соответствий, интегрирующие точки зрения;
- условия применимости;
- согласованность фреймворка с положениями концептуальной модели стандарта ISO/IEC/IEEE 42010.

В целом архитектурный фреймворк устанавливает общие практики создания, интерпретации, анализа и использования AD в рамках конкретной области

применения или сообщества заинтересованных сторон. Примеры архитектурных фреймворков: MODAF, TOGAF, Kruchten's 4+1 View Model, RM-ODP. Эти, а также множество других архитектурных фреймворков были учтены при разработке архитектурного фреймворка стандарта ISO/IEC/IEEE 42010.

Концептуальная модель архитектурного фреймворка (Architecture Framework) приведена на рис. 1.5 [53].

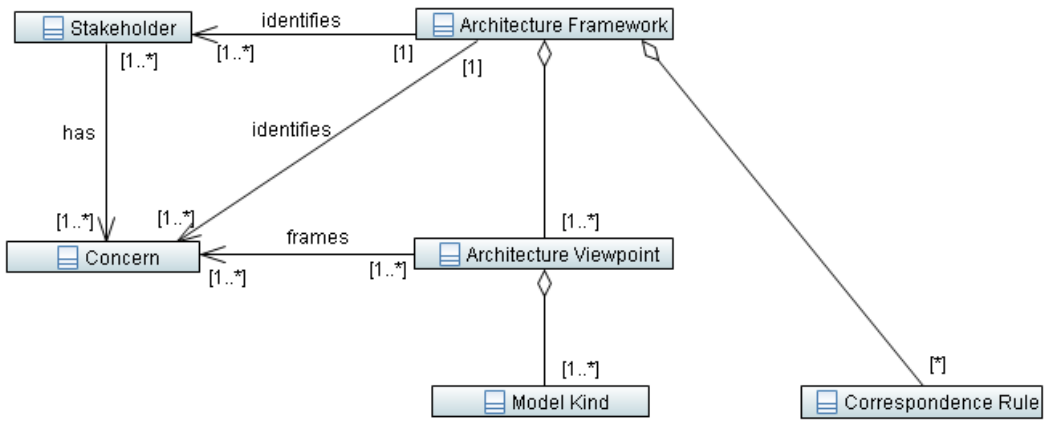

Рисунок 1.5 – Диаграмма классов, описывающая архитектурный фреймворк

Языком описания архитектуры является любая форма выражения, которая может быть использована в описании архитектуры. Язык описания архитектуры может быть основан на конкретном виде моделей, точке зрения, или на нескольких точках зрения одновременно. Примеры языков описания архитектуры: Rapide, SysML, ArchiMate, ACME, xADL.

Концептуальная модель языков описания архитектуры (Architecture Description Language) приведена на рис. 1.6 [53].

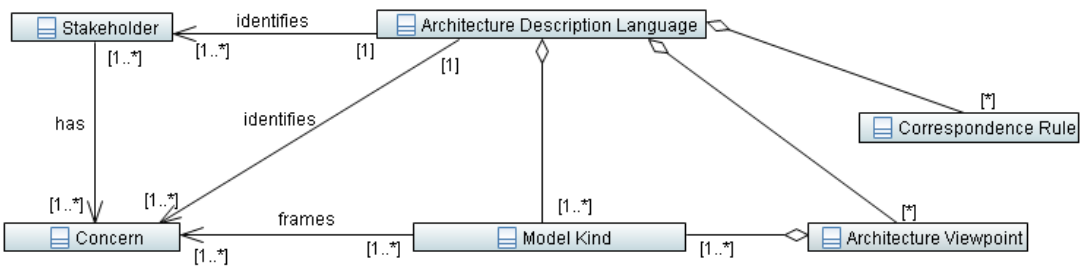

Рисунок 1.6 – Диаграмма классов, описывающая взаимосвязь основных понятий языка описания архитектуры

Подводя итоги анализа развития понятия «Архитектура системы», можно сделать следующий вывод: в отличие от академического интереса прошлых лет,

современный интерес к тому, что скрывается за понятием «архитектура системы», приобретает отчетливый прикладной характер. Желание снизить непроизводительные затраты на покупку и внедрение отдельных элементов ИС, представление подразделений по сопровождению и эксплуатации ИС на предприятиях как центров затрат и стремление руководства исключить эти «лишние» затраты из совокупности затрат на осуществление управления БП предприятия – все эти и целый ряд других факторов определяет необходимость решения проблемы архитектурной интеграции разнородных элементов ИС на формальном уровне, доступном для реализации в виде отдельного компонента современных ИС. Такой компонент позволит сократить затраты различных ресурсов (прежде всего, затраты времени) на внедрение и эксплуатацию отдельных функциональных элементов ИС за счет автоматизации выполнения операций интеграции, поиска и устранения противоречий между этими элементами.

1.4 Требования к информационной системе: основные определения

Как отмечено выше, проблема формирования такого варианта конфигурации ФС ИС, который позволил бы формировать и отображать единое целостное информационное представление объекта или процесса в соответствии с поставленными перед ИС целями, в настоящее время не имеет объективного решения. Предлагаемые отдельными специалистами частные решения основаны, главным образом, на интуиции и опыте этих специалистов и не могут использоваться для других ИС без коренного преобразования. Объективным решение данной проблемы будет в том случае, если удастся сформировать его математическое описание, справедливое для большинства ИС[7].

Однако для создания математического описания законов, процессов и методик формирования эффективных и качественных вариантов конфигураций ФС ИС из множества типовых ФЗ или ИТ-услуг необходимо, прежде всего, рассмотреть основные источники информации, которые определяют способы представления этих ФЗ или ИТ-услуг. Такими источниками являются требования, которые выдвигаются к ИС, ее отдельным ФЗ или ИТ-услугам, а также к видам обеспечений ИС руководством и сотрудниками автоматизируемого объекта или процесса. В связи с этим возникает необходимость рассмотреть основные определения понятия «требование к ИС», а также проанализировать основные подходы к анализу и управлению такими требованиями.

Так же, как и в случае определения понятия «информационная система», начнем анализ определений понятия «требование к информационной системе» с изучения основных стандартов. В ГОСТах группы 34 «Информационные тех-

[7] Примером такого математического описания объективного решения проблемы в компьютерных науках является реляционная модель данных, предложенная Э. Коддом. Эта модель является, фактически, универсальным математическим аппаратом, позволяющим за счет некоторой избыточности описать и, следовательно, управлять практически любой реляционной или объектно-реляционной базой данных и соответствующей СУБД.

нологии» термин «требования» до сих пор не имеет специального определения. Однако данный термин встречается в описаниях содержания документов с результатами трех первых стадий создания автоматизированной системы (АС) – «Формирование требований к АС», «Разработка концепции АС» и «Техническое задание» [71, 72].

В британской методологии разработки ИС SSADM требование к ИС определяется, в частности, как ответ на следующие вопросы [73, 74]:

- «Что требуется от системы?»;
- «Зачем это нужно?»;
- «Кому это необходимо?»;
- «Насколько это важно?»;
- «Чем можно измерить степень соблюдения требования?».

Более развито понимание термина «требование» применительно к разработке обеспечивающей части ИС, а именно программного обеспечения. Стоит отметить, что уже достаточно давно, говоря о требованиях к ИС, на самом деле имеют в виду требования к программному обеспечению этой ИС. Однако данная подмена понятий не является бесспорной. Поэтому рассмотрим определения требований, сложившиеся в программной инженерии.

В стандарте разработки программного обеспечения IEEE 610.12-1990 понятие ***«требование»*** имеет следующие определения [75]:

- условие или возможность, необходимые пользователю для решения проблемы или достижения цели;

- условие или возможность, которыми должна обладать система или компонент системы, соответствующие договору, стандарту, спецификации или другому официальному документу;

- документированное представление условия или возможности подобно описанному в первых двух определениях.

Данные определения до сих пор считаются рядом специалистов основным определением понятия «требование» в программной инженерии. Так, например, Д. Леффингуэлл и Д. Уидриг в своем труде [76] отмечают, что данные определения, точнее – два первых из них, являются вполне приемлемыми общими определениями понятия «требование».

С другой стороны, К.И. Виггерс в своей книге [77] отмечает, что отсутствие общепринятых определений требований, которыми пользуются для описания выполняемой работы, является одной из проблем индустрии программного обеспечения: «Разные эксперты, говоря об одном и том же документе, называют его и требования пользователя, и требования к программному обеспечению, и функциональные требования, и системные требования, и технологические требования, и бизнес-требования, и требования к продукту. Заказчики зачастую считают, что требования – это развитая концепция продукта, предназначенная для разработчиков. Те, в свою очередь, полагают, что в отношении клиентов это детальная разработка интерфейса пользователя. Такое многообразие ведет к сумятице и раздражающим проблемам коммуникации».

При этом К.И. Виггерс наряду с упомянутым выше определением требования по IEEE 610.12-1990 приводит еще два следующих определения требований:

а) требование – это нечто такое, что приводит к выбору дизайна [78];

б) требования – это спецификация того, что должно быть реализовано. В них описано поведение системы, свойства системы или ее атрибуты. Они могут быть ограничены процессом разработки системы [79].

С другой стороны, практически все стандарты, методологии, технологии, а также теоретические работы и практические исследования, посвященные проблеме формирования и управления требованиями, отмечают необходимость разделения множества всех возможных требований к ИС (или же к программной системе) на ряд отдельных подмножеств требований, имеющих некие общие признаки. Так, например, в ГОСТах группы 34 «Информационные технологии» на первой стадии разработки АС – «Формирование требований к автоматизированной системе» – требования к АС разделяются на две основные группы [71]:

а) требования к характеристикам реализации функций и задач в соответствии с действующими нормативно-техническими документами, определяющими общие технические требования к АС конкретного вида;

б) дополнительные требования к АС в целом и ее частям, учитывающие специфику создаваемой АС.

В дополнение к этим группам требований в документе, описывающем результаты выполнения работ на стадии «Формирование требований к автоматизированной системе», выделяют ряд общих рекомендаций следующего характера [71]:

а) по виду создаваемой АС, ее совместимости с другими АС и неавтоматизируемой частью соответствующей системы;

б) по организационной и функциональной структуре создаваемой АС;

в) по составу и характеристикам подсистем и видов обеспечений АС;

г) по организации использования имеющихся и приобретению дополнительных средств вычислительной техники;

д) по рациональной организации разработки и внедрения АС;

е) по определению основных и дополнительных, внешних и внутренних источников и видов объемов финансирования и материального обеспечения разработок АС;

ж) по обеспечению производственных условий создания АС;

и) другие рекомендации по созданию АС.

Данные рекомендации являются, по сути, слабо формализованными описаниями требований к АС, которые на данной стадии сформулированы недостаточно четко.

В документе, содержащем описание результатов выполнения работ на стадии «Разработка концепции автоматизированной системы», приводится сопоставительный анализ требований пользователя к АС и вариантов концепции АС на предмет удовлетворения требованиям пользователя. Кроме того, в данном документе приводятся описания требований, гарантирующих качество АС [71].

В ГОСТ 34.602-89 «Техническое задание на создание автоматизированной системы» требования к АС делятся на три основные группы [72]:

а) требования к системе в целом:

б) требования к функциям (задачам), выполняемым системой;

в) требования к видам обеспечений.

Группа требований к системе в целом, в свою очередь, делится на следующие виды требований: требования к структуре и функционированию системы; требования к численности и квалификации персонала системы и режиму его работы; показатели назначения; требования к надежности; требования безопасности; требования к эргономике и технической эстетике; требования к транспортабельности для подвижных АС; требования к эксплуатации, техническому обслуживанию, ремонту и хранению компонентов системы; требования к защите информации от несанкционированного доступа; требования по сохранности информации при авариях; требования к защите от влияния внешних воздействий; требования по стандартизации и унификации; дополнительные требования.

Группа требований к видам обеспечений, в свою очередь, делится на следующие требования (в зависимости от вида системы): к математическому, информационному, лингвистическому, программному, техническому, метрологическому, организационному, методическому и другим видам обеспечений.

В методологии SSADM существует деление требований к ИС на функциональные и нефункциональные. Функциональные требования определяют, что должна делать создаваемая система. Зачастую функциональные требования формируются на основе выявленных ранее бизнес-правил – правил и закономерностей выполнения элементов автоматизируемых бизнес-процессов предприятия. Нефункциональные требования в общем случае могут быть самыми разнообразными. Обычно выделяют следующие типы нефункциональных требований [73, 74]:

- требования к качеству выполнения функций;
- требования к ограничению доступа;
- требование к безопасности;
- требования к обеспечению мониторинга и контролируемости;
- требования к ограничениям;
- архитектурные требования и т.д.

Д. Леффингуэлл и Д. Уидриг предполагают разделение требований на следующие группы [76]:

а) область проблемы (технические или бизнес-задачи предметной области);

б) потребности заинтересованных лиц, в том числе пользователей системы (под термином «потребность» Д. Леффингуэлл и Д. Уидриг понимают отражение некоей личной, рабочей или бизнес-проблемы (или возможности), решение которой оправдывает замысел, покупку или использование новой системы);

в) функции системы (под термином «функция» Д. Леффингуэлл и Д. Уидриг понимают предоставляемое системой обслуживание для удовлетворения одной или нескольких потребностей);

г) требования к программному обеспечению, а именно:

- функциональные требования к программному обеспечению;
- нефункциональные требования к программному обеспечению (практичность, надежность, производительность, возможность обслуживания);
- ограничения проектирования.

Для описания последовательности действий, выполняемых системой, чтобы предоставить значимый результат пользователю, Д. Леффингуэлл и Д. Уидриг предлагают использовать прецеденты (use cases).

К.И. Виггерс, в свою очередь, предлагает разделение требований на следующие группы, которые он называет «уровнями» [77]:

а) бизнес-требования, которые содержат высокоуровневые цели организации или заказчиков системы;

б) требования пользователей, описывающие цели и задачи, которые пользователям позволит решить система;

в) функциональные требования, определяющие функциональность ПО, которое разработчики должны построить, чтобы пользователи смогли выполнить свои задачи в рамках бизнес-требований;

г) системные требования – высокоуровневые требования к системе, способные определять функциональные требования.

К.И. Виггерс также предлагает при документировании функциональных требований выделить нефункциональные требования, описывающие цели и атрибуты качества функций системы.

Организация уровней требований, информационная взаимосвязь между отдельными уровнями, а также ограничения, возникающие в процессе формирования требований отдельных уровней, показаны на рис. 1.7 [77]. На данной схеме овалами обозначены типы информации для требований, а прямоугольниками – способы хранения информации (документы, диаграммы, базы данных).

Сотрудники компании Telelogic, разрабатывающей пакеты управления требованиями Telelogic DOORS, предлагают разделение требований к системе на следующие группы [80]:

а) формулировки потребностей;

б) пользовательские требования;

в) системные требования;

г) требования для компонентов системы;

д) требования для компонентов подсистем.

Взаимосвязь этих групп в процессе разработки системы показана на рис. 1.8 [80]. Здесь овалами обозначены выполняемые процессы, а прямоугольниками – данные, которые используются для выполнения процессов или являются результатами выполнения этих процессов.

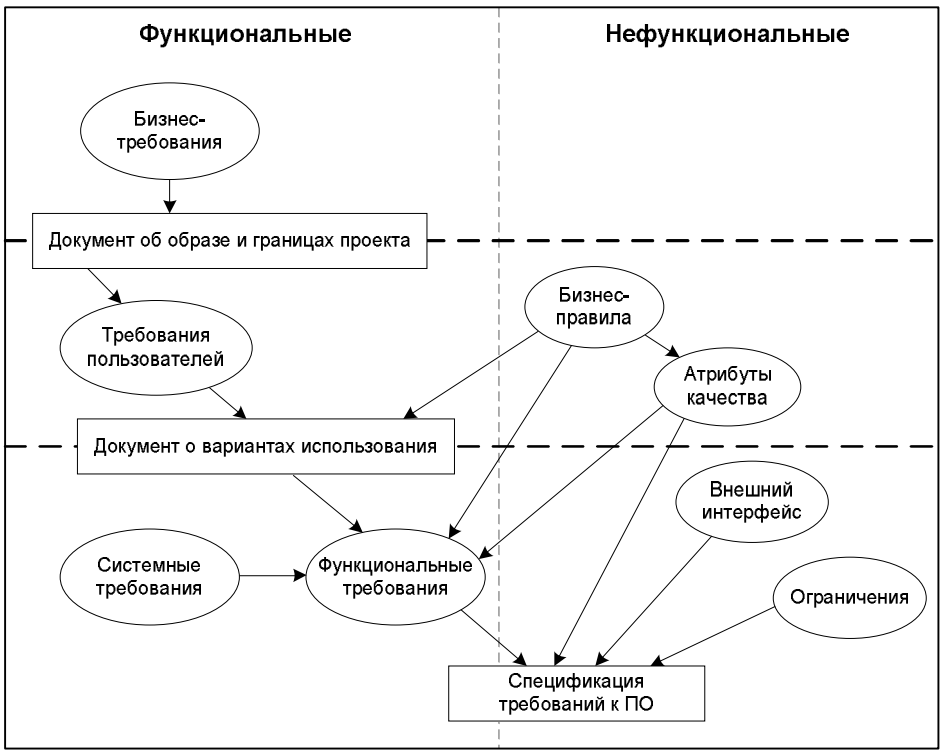

Рисунок 1.7 – Взаимосвязи уровней требований к системе по К.И. Виггерсу

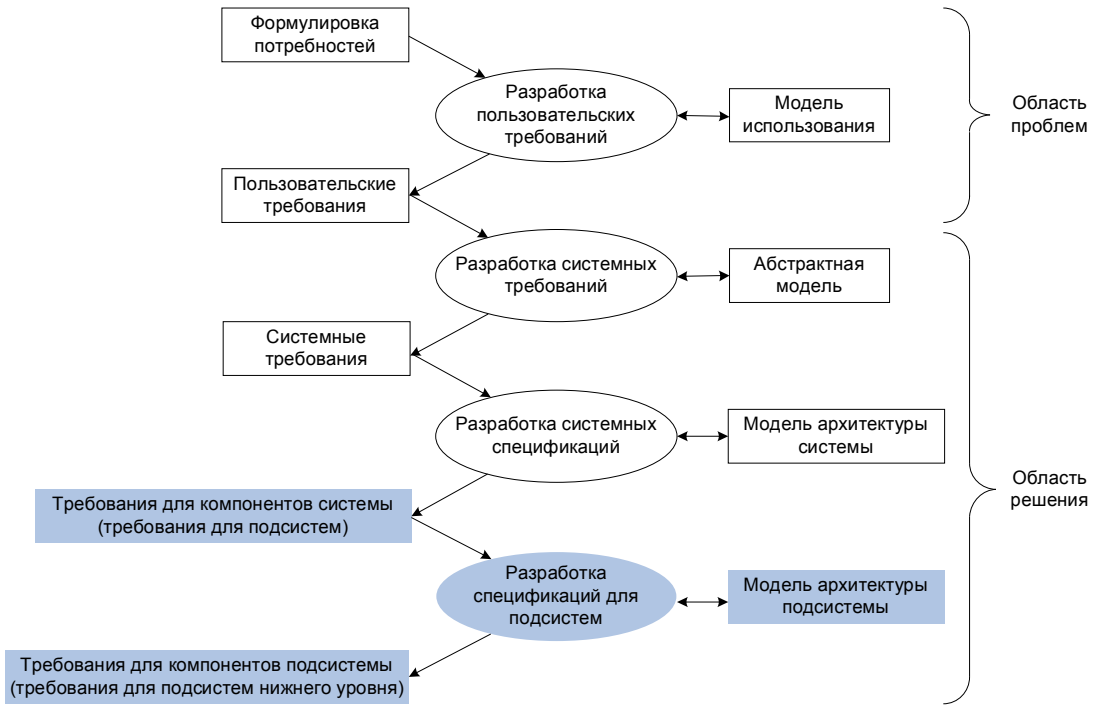

Рисунок 1.8 – Взаимосвязь групп требований в процессе разработки системы по версии компании Telelogic

Рассмотренные попытки определений понятий «требование» и «требование к ИС» показывают, что выделить единое определение данных понятий и придерживаться его очень трудно. Ряд специалистов придерживается опреде-

ления понятия «требование», изложенного в стандарте IEEE 610.12-1990. В то же время другие специалисты пользуются теми определениями, которые наилучшим образом отражают особенности предлагаемых этими специалистами методов и методик формирования, анализа и управления этими требованиями. Аналогичным образом обстоит ситуация и с разделением требований на группы по неким общим признакам. Следует отметить, что к настоящему времени классификация подобных признаков не получила общепринятого научного обоснования.

1.5 Основные подходы к описанию требований

Как показано в предыдущем подразделе, выделить единое устоявшееся определение понятия «требование к ИС» очень трудно. Даже в области программной инженерии, которая на текущий момент получила наибольшее развитие как за рубежом, так и на Украине, данная проблема, несмотря на наличие стандартизованного определения понятия «требование», остается актуальной. Своеобразным решением этой проблемы являются попытки уточнения определений понятий «требование», «требование к ИС» и рассмотренных выше групп требований с помощью их атрибутивных описаний.

По сути такое уточнение определений общего понятия «требование» и отдельных групп требований, выделяемых из общего понятия на основе какого-либо общего признака, является иллюстрацией принципа двойственности, предложенного Ю. Шрейдером и А. Шаровым. Данный принцип обеспечивает согласованность двух методов оценки сходства – по таксономии и мерономии: «Чем больше классификатор знает о реальной структуре таксонов, тем точнее он может определить архетип и осуществить гомологии. И, наоборот, более точное определение архетипов позволяет гораздо точнее определить таксономическую структуру» [81].

На практике подобные уточнения воплощаются в виде атрибутивных описаний требований, которые, однако, также не являются общепринятыми и различаются в зависимости от их авторства. Так, анализ описания требований к функциям (задачам), выполняемым системой (согласно ГОСТ 34.602-89), показывает использование для такого описания следующей неявной атрибутивной модели [72]:

а) перечень функций, задач или их комплексов (в том числе обеспечивающих взаимодействие частей системы), подлежащих автоматизации;

б) перечень функциональных подсистем, отдельных функций или задач, вводимых в действие в первой и последующих очередях (при создании системы в две или более очереди);

в) временной регламент реализации каждой функции, задачи (или комплекса задач);

г) качество реализации каждой функции (задачи или комплекса задач);

д) форма представления выходной информации;

е) характеристики необходимой точности и времени выполнения, требования одновременности выполнения группы функций, достоверности выдачи результатов;

ж) перечень и критерии отказов для каждой функции, по которой задаются требования по надежности.

Данная модель названа неявной потому, что в указанном стандарте, да и в других стандартах группы 34 «Информационные технологии» отсутствует система показателей или хотя бы множество атрибутов, которые позволяют унифицировать описания требований к функциям (задачам), выполняемым системой. Вместе с тем, в указанном стандарте данная модель представлена в виде рекомендаций по содержанию соответствующего подраздела документа «Техническое задание на АС».

В методологии SSADM описания требований к ИС более структурированы, чем в рассмотренном выше ГОСТе. При этом каждое описание функционального требования может быть дополнено описанием множества нефункциональных требований. В общем случае, атрибутивные модели требований в методологии SSADM будут иметь следующий вид [74]:

а) характеристики требования в каталоге требований:

- кодовое обозначение проекта;
- фамилия и инициалы разработчика требования;
- дата формирования требования;
- стадия, на которой находится разработка ИС в соответствии с положениями методологии SSADM;
- версия описания требования;
- номер страницы описания требования;
- количество страниц описания требования;

б) атрибутивное описание функционального требования:

- источник требования;
- важность требования;
- ответственный за выполнение требуемой функции;
- имя функционального требования;
- описание функционального требования;
- преимущества, возникающие в результате выполнения функционального требования;

в) атрибутивное описание нефункционального требования:

- описание нефункционального требования;
- целевое значение нефункционального требования;
- допустимое значение нефункционального требования;
- примечание;

г) описание результатов выполнения требования:

- предлагаемое решение требования;

- ссылка на документы;
- ссылка на другие требования;
- резолюция (ссылка на функцию, реализующую требование).

Следует отметить, что в методологии SSADM требования к системе в целом практически не рассматриваются и не описываются в виде отдельных требований.

Таким образом, существующие на текущий момент атрибутивные модели требований к ИС, определяемые отечественными ГОСТами и зарубежными методологиями разработки ИС, достаточно ограничены и ориентированы преимущественно на описание функциональных требований.

Атрибутивные модели требований к ПО несколько разнообразней. Так, Д. Леффингуэлл и Д. Уидриг предлагают для группы требований, описывающих область проблемы, выделять все стороны, заинтересованные в решении некоей бизнес-проблемы автоматизированным способом. Каждая из этих сторон выдвигает свои требования, описывающие область проблем с помощью следующего шаблона ключевых слов [76]:

- «Проблема» (описание возникающей в бизнесе проблемы, для решения которой предполагается разработка ПО);
- «воздействует на» (перечень организаций, подразделений и отдельных лиц конкретной заинтересованной стороны, которые участвуют в решении описанной выше проблемы);
- «результатом чего» (описание недостатков, возникающих вследствие существования описанной выше проблемы);
- «выигрыш от» (описание источников выигрыша и перечисление количественных и качественных результатов выигрыша, возникающих вследствие применения данного источника).

Для описания группы требований, отражающих потребности заинтересованных лиц, авторы работы [76] вообще не предлагают специальные атрибутивные модели, замечая, что такие потребности будут неоднозначными и размытыми. При этом для описания группы требований, отражающих функции системы, предлагается использовать следующее множество атрибутов, которые, согласно [76], являются общеупотребительными и применяются в большинстве проектов:

- статус;
- приоритет/полезность;
- трудоемкость;
- риск;
- стабильность;
- целевая версия продукта;
- назначение;
- обоснование.

Для описания функциональных требований к ПО Д. Леффингуэлл и Д. Уидриг предлагают использовать простые декларативные определения или же

прецеденты (use cases) языка UML, а для описания нефункциональных требований к ПО они предлагают не готовые атрибутивные модели, а рекомендации по использованию тех или иных показателей. Так, для описания требований к практичности рекомендуется использовать такие показатели [76]:

- необходимое время подготовки пользователя для достижения минимальной и операционной производительности;
- время выполнения типичных задач или транзакций, осуществляемых конечным пользователем;
- сравнение практичности новой системы с уже существующими современными системами, которые известны и пользуются успехом у пользователей;
- указание на существование различных форм документации и помощи, а также определение необходимых функций этих форм;
- указания на соответствия соглашениям и стандартам, разработанным для человеко-машинного интерфейса.

Для описания требований к надежности рекомендуется использовать такие показатели [76]:

- доступность;
- среднее время между отказами;
- среднее время восстановления;
- точность;
- максимальный коэффициент ошибок;
- количество различных ошибок.

Для описания требований к производительности рекомендуется использовать такие показатели:

- время ответа для транзакции (среднее, максимальное);
- пропускная способность (число транзакций в секунду);
- емкость (сколько пользователей или транзакций может обслужить система);
- режимы снижения производительности (допустимые режимы работы при ухудшении параметров системы).

Но для описания требований к возможности обслуживания и ограничений проектирования конкретные атрибуты или показатели не рекомендуются [76].

К.И. Виггерс предлагает несколько иное описание выделенных им групп требований. Так, описания бизнес-требований предлагается осуществлять без использования атрибутивной модели описания. Структуризация описания бизнес-требований и создаваемой на их основе системы осуществляется путем создания документа об образе и границах проекта, шаблон которого имеет следующую структуру [77]:

а) бизнес-требования:

- исходные данные;
- возможности бизнеса;
- бизнес-цели и критерии успеха;
- потребности клиента или рынка;
- бизнес-риски;

б) образ решения:

- положение об образе проекта;
- основные функции;
- предположения и зависимости;

в) масштабы и ограничения проекта:

- объем первоначально запланированной версии;
- объем последующих версий;
- ограничения и исключения;

г) бизнес-контекст:

- профили заинтересованных лиц;
- приоритеты проекта;
- операционная среда.

При этом для создания образа решения предлагается использовать шаблон ключевых слов, предложенный в работе [82], который частично совпадает с рассмотренным выше шаблоном, предложенным в [76] и имеет следующую структуру:

- «для» (целевая аудитория покупателей);
- «который» (положение о потребностях или возможностях);
- «эта (этот)» (имя продукта);
- «является» (категория продукта);
- «который» (ключевое преимущество, основная причина для покупки или использования);
- «в отличие от» (основной конкурирующий продукт, текущая система или текущий бизнес-процесс);
- «наш продукт» (положение об основном отличии и преимуществе нового продукта).

Для описания основных функций нового продукта предлагается использовать минимальное количество атрибутов, в частности, уникальное имя или обозначение каждой функции.

Для описания профилей заинтересованных лиц предлагается использовать следующую атрибутивную модель [77]:

а) основная ценность или преимущество, которое продукт принесет заинтересованным лицам и то, как продукт удовлетворит покупателей;

б) вероятное отношение заинтересованных лиц к продукту;

в) наиболее интересные для заинтересованных лиц функции и характеристики продукта;

г) все известные ограничения, которые, по мнению заинтересованных лиц, должны быть соблюдены.

Для выявления требований пользователей рекомендуется применять варианты использования, которые могут быть описаны с помощью следующей атрибутивной модели [77]:

- идентификатор;

- имя, кратко описывающее задачи пользователя в формате «глагол + объект», например, «разместить заказ»;
- краткое текстовое описание варианта использования на естественном языке;
- список предварительных условий, которые должны быть удовлетворены до начала разработки варианта использования;
- пронумерованный список действий, иллюстрирующий последовательность этапов взаимодействия лица и системы от предварительных условий до выходных условий.

В процессе документирования вариантов использования эта атрибутивная модель расширяется до полного шаблона варианта использования, предложенного Алистером Коберном, который выглядит следующим образом [83]:

а) характеристики варианта использования:
- идентификатор;
- название варианта использования;
- автор варианта использования;
- дата создания варианта использования;
- автор последнего обновления варианта использования;
- дата последнего обновления варианта использования;

б) характеристики действующего лица:
- наименование;
- описание;
- предварительные условия;
- выходные условия;

в) характеристики варианта использования:
- нормальное направление развития варианта использования;
- альтернативное направление развития варианта использования;
- исключение;
- включение;
- приоритет;
- частота использования;
- бизнес-правила;
- специальные требования;
- предположения;
- замечания и вопросы.

Однако такая подробная атрибутивная модель используется далеко не всегда. Поэтому в [83] предлагается использовать не только полный, но и рабочий шаблон вариантов использования, который включает в себя текстовое изложение целей пользователя и взаимодействия пользователя с системой.

Для описания функциональных требований рекомендуется использовать один из сложившихся шаблонов спецификации требований к программному обеспечению, основанный на идеях стандарта IEEE Standard 830-1998. Такой шаблон годится для самых разных проектов, однако он не лишен ограничений в

использовании и неясных мест. Развитие этого шаблона представляет собой документ со следующей структурой [77]:

а) введение:
- назначение проекта;
- соглашения, принятые документы;
- предполагаемая аудитория и рекомендации по чтению;
- границы проекта;
- ссылки;

б) общее описание:
- общий взгляд на продукт;
- особенности продукта;
- классы и характеристики пользователей;
- операционная среда;
- ограничения дизайна и реализации;
- документация для пользователей;
- предположения и зависимости;

в) функции системы:
- название функции системы;
- описание и приоритеты функции системы;
- последовательности «воздействие – реакция»;
- функциональные требования;

г) требования к внешнему интерфейсу:
- интерфейсы пользователя;
- интерфейсы оборудования;
- интерфейсы ПО;
- интерфейсы передачи информации;

д) другие нефункциональные требования:
- требования к производительности;
- требования к охране труда;
- требования к безопасности;
- атрибуты качества;

е) остальные требования;

ж) словарь терминов;

и) модели анализа требований;

к) список вопросов.

Данный документ следует дополнять визуальными моделями, отражающими представление требований более формальными средствами, чем естественный язык или же его варианты. К таким моделям относятся:
- диаграммы потоков данных;
- диаграммы «сущность – связь»;
- диаграммы перехода состояний;
- карты диалогов;
- диаграммы вариантов использования;

- диаграммы классов;
- диаграммы взаимодействия;
- другие нестандартные диаграммы (при необходимости).

В то же время другие исследователи [84] считают, что при описании требований К.И. Виггерс применяет следующую атрибутивную модель:

- дата создания требования;
- номер его текущей версии;
- автор требования;
- лицо, ответственное за удовлетворение требования;
- ответственный за требование или список заинтересованных лиц (чтобы принимать решения о предложенных изменениях);
- состояние требования;
- происхождение или источник требования;
- логическое обоснование требования;
- подсистема (или подсистемы), для которых предназначено требование;
- номер версии продукта, для которого предназначено требование;
- используемый метод проверки или критерий тестирования приемлемости;
- приоритет реализации;
- стабильность требования.

С точки зрения английского подразделения INCOSE, требование следует описывать набором категорий атрибутов [80], каждая из которых состоит из различного числа атрибутов. Такой набор имеет следующий вид:

а) идентификация:
- идентификатор требования;
- класс требования;

б) внутренние характеристики:
- основной тип;
- качественный подтип;
- тип продукта/процесса;
- количественный/качественный тип;
- фаза жизненного цикла;

в) приоритет и важность:
- приоритет;
- важность;

г) источник и владелец:
- способ получения;
- источник;
- владелец;
- согласовано;

д) контекст:
- набор требований/документ;
- объект;
- границы (рамки);

е) проверка и утверждение (верификация и валидация, V&V):

- V&V метод;
- V&V стадия;
- V&V статус;
- критерий успешности проверки;
- критерий утверждения;

ж) поддержка процесса:

- статус согласования;
- статус проверки;
- статус удовлетворения;
- статус рецензирования;

и) уточнение:

- необходимость;
- комментарии;
- вопросы;
- ответы;

к) прочее:

- зрелость (стабильность);
- уровень риска;
- оценочная стоимость;
- фактическая стоимость;
- релиз продукта.

Таким образом, исследование атрибутивных описаний требований к ИС, требований к ПО как более широкого класса требований, а также изучение документов, используемых для описания и спецификации подобных требований позволяет утверждать следующее:

а) не существует единой атрибутивной модели описания требования;

б) атрибутивные модели описания требований могут уточняться, изменяться и дополняться новыми атрибутами в зависимости от особенностей разрабатываемой системы, предметной области, заинтересованных сторон и других факторов;

в) описания бизнес-требований, потребностей пользователей и аналогичных по назначению групп требований носят практически неформализованный характер и очень слабо используют атрибутивные модели;

г) наиболее формальны описания функциональных требований к ПО, которые основаны на атрибутивных моделях и совместном использовании формального аппарата той или иной разновидности визуальных моделей.

1.6 Основные подходы к организации управления требованиями

Описание требований через создание и развитие их атрибутивных моделей определяет ***основной принцип управления требованиями – постепенное преобразование множества начальных значений атрибутов, описывающих требование, в множество желаемых значений тех же атрибутов. Под желаемым значением следует понимать значение, которое приобретает атрибут при описании реализованного требования, проверенного соответствующими тестами.***

В соответствии с этим принципом организованы практически все существующие подходы к управлению требованиями в ИТ-сфере. При этом следует отметить, что в ГОСТах группы 34 «Информационные технологии» проблема управления требованиями не рассматривается, а рекомендации по организации и документированию работ по управлению требованиями к ИС отсутствуют.

Д. Леффингуэлл и Д. Уидриг рассматривают управление требованиями как систематический подход к выявлению, организации и документированию требований к системе, а также процесс, в ходе которого вырабатывается и обеспечивается соглашение между заказчиком и выполняющей проект группой по поводу меняющихся требований к системе [76]. Подобную точку зрения занимает и К.И. Виггерс, определяя понятие «управление требованиями» как выработку и поддержание взаимного согласия с заказчиками по поводу требований к разрабатываемому ПО [85], в ходе которых должны выполняться такие действия [77]:

- определение основной версии требований (моментальный срез требований для конкретной версии продукта);
- просмотр предлагаемых изменений требований и оценка вероятности воздействия каждого изменения до его принятия;
- включение одобренных изменений требований в проект установленным способом;
- согласование плана проекта с требованиями;
- обсуждение новых обязательств, основанных на оцененном влиянии изменения требований;
- отслеживание отдельных требований до их дизайна, исходного кода и вариантов тестирования;
- отслеживание на протяжении всего проекта статуса требования и действий по изменению этого статуса.

Для выполнения последнего действия из этого списка он предлагает использовать следующий шаблон определения статуса требования [77] (см. табл. 1.1).

Согласно методологии RUP, управление требованиями – это систематический подход к выявлению, организации и документированию требований к системе, а также установка и поддержание соглашения между клиентом и группой

Таблица 1.1 — Шаблон определения статуса требования

Состояние	Определение
Proposed (Предложено)	Требование запрошено авторизованым источником
Approved (Одобрено)	Требование проанализировано, его влияние на проект просчитано, требование было размещено в базовой версии проекта. Ключевые заинтересованные в проекте лица согласились с этим требованием, а разработчики ПО обязались реализовать его
Implemented (Реализовано)	Код, реализующий требование, разработан, написан и протестирован. Требование отслежено до соответствующих элементов дизайна и кода
Verified (Проверено)	Корректное функционирование реализованного требования подтверждено в соответствующем продукте. Требование отслежено до соответствующих вариантов тестирования. Теперь требование считается завершенным
Deleted (Удалено)	Утвержденное требование удалено из базовой версии. Необходимо описать причины удаления и назвать того, кто принял решение об удалении требования
Rejected (Отклонено)	Требование предложено, но не запланировано для реализации ни в одной из будущих версий. Необходимо описать причины отклонения и назвать того, кто принял это решение

разработки по поводу изменений требований к системе. Данное соглашение, как и тексты исходных требований, подлежит документальному оформлению. В общем случае, процесс управления требованиями, согласно RUP, разделяется на следующие работы [86]:

- разработка плана управления требованиями;
- разработка концепции разрабатываемой системы;
- построение и детализация моделей сценариев использования разрабатываемой системы.

Следует отметить, что в настоящее время процессное представление работ, в том числе – работ по управлению требованиями, является основным представлением работ по созданию и модификации ПО.

В основе подавляющего большинства инструментальных средств управления требованиями положена идея централизованного репозитория требований. Такой репозиторий чаще всего представляет собой иерархические структуры данных, в которых на верхнем уровне находятся требования к системе в целом.

Эти требования (что должна делать система, как должна работать система) детализируются через формулирование более мелких требований, уточняющих отдельные аспекты требований к системе в целом. Аналогичным образом реализована и модель изменений требований.

1.7 Опыт управления требованиями: отличия от теории

В подразд. 1.4 – 1.6 были рассмотрены основные подходы к определению понятия «требование», к описанию и организации управления требованиями. Однако следует постоянно помнить, что специфика управления требованиями в каждой конкретной организации в процессе создания конкретных ИС может довольно сильно отличаться от приводимых в литературе общих методов формирования и управления требованиями.

Это понимают и специалисты, занимающиеся разработкой новых, в том числе – формальных методов формирования, анализа и управления требованиями. Так, например, в [80] отмечается: «Управление требованиями… является широко распространенным и привычным термином, однако, несмотря на это, для большей части людей он до сих пор остается непонятным; другая же часть и вовсе понимает его неправильно. Это уже само по себе является причиной того, что требования бывают очень часто плохо написаны, а значит, и плохо управляются. Возрастающий на организацию прессинг современных условий существования и конкуренции является основной причиной отказа ее руководства от внедрения более четких процедур в области управления требованиями. При этом совершенно упускается из виду то, что управление требованиями может помочь организации работать более эффективно». Иными словами: с точки зрения авторов многих работ, в которых освещаются общие вопросы по формированию и управлению требованиями к ИС, ИТ или любой другой разновидности ПО, большинство ошибок, связанных с требованиями и процессом управления этими требованиями:

а) лежит на совести исполнителей этих работ и их руководства;

б) возникают вследствие низкой квалификации исполнителей работ и их руководства;

в) являются последствиями желания выполнить работы по созданию ПО как можно быстрее.

В то же время специалисты-практики, которые непосредственно участвуют в выполнении работ с требованиями, высказывают несколько иное мнение о причинах и природе возникающих ошибок. В качестве примера подобной точки зрения, возникшей в ходе практического решения задач управления требованиями, рассмотрим результаты выполнения работ по формированию и управлению требованиями одной из организаций-разработчиков ИС, изложенные в

статье Ю.О. Волкова «Управление требованиями и автоматизация этого процесса» [87].

Во-первых, требования к разрабатываемой ИС, существовавшие на момент начала работы, были очень общими, а рамки проекта – в значительной мере неопределёнными. Процессы управления требованиями были усложнены также из-за отсутствия единого представителя заказчика, уполномоченного принимать решения по вопросам, возникающим в ходе сбора требований.

Во-вторых, разработчикам ИС пришлось самостоятельно дорабатывать методологию сбора требований, поскольку сложность проекта не позволяла использовать готовые методологии, имеющие существенные ограничения. В частности, отмечается: «… в книге Д. Леффингуэлла [76] … за описанием методов анализа систем и управления требованиями следует конкретный пример методологии, предваряемый описанием ограничений его применения. Так вот, практически каждый пункт этого списка … относится к нашему проекту, т.е. приведённая методология в буквальном смысле не подходит нам «по всем пунктам…». Поэтому процесс сбора и управления требованиями стал композицией процесса сбора требований с помощью «вариантов использования» (use cases)[8], предложенного А. Кобёрном [83], а также упрощённого унифицированного процесса фирмы Rational [56]. Для выработки унифицированного понимания этого процесса разработчики создали специальный документ «Концепция управления требованиями».

В-третьих, с самого начала работы с требованиями к ИС разработчиками была сделана ставка на то, что использование инструментального средства автоматизации управления требованиями позволит автоматически «собрать» готовый документ «Техническое задание» (ТЗ) на основе информации репозитория. Причина такого целеполагания заключалась в опасениях разработчиков, что вручную перенос требований в ТЗ займёт слишком много времени, поэтому параллельно со сбором требований дорабатывался соответствующий шаблон ТЗ (в формате MS Word), содержащий скрипт для автоматической генерации ТЗ. В качестве такого инструментального средства был выбран продукт IBM Rational RequisitePro 2003. Этот выбор был сделан практически без анализа альтернатив – по словам Ю.О. Волкова: «… на такой анализ просто не было времени, рисковать не хотелось, а фирма Rational себя уже давно зарекомендовала, в том числе и в области методологии». Поэтому разработчики начали работать с пакетом RequisitePro, надеясь, что возникающие проблемы будут решаться «по ходу дела». Общим результатом работы и взаимодействия людей, занимающихся сбором требований, стал репозиторий требований RequisitePro, а основным способом доступа к этому репозиторию – Web-интерфейс.

8 Согласно Ю.О. Волкову, «варианты использования» содержат описания сценариев и другие атрибуты («основное действующее лицо», «цель» и т.п.), а «требование – это нечто более простое, например, фраза, описывающая это требование.

Однако проведенный выбор инструментального средства автоматизации управления требованиями существенно ограничил выполнение процесса сбора и управления требованиями. Так, например, выбранный пакет [87]:

а) не позволяет объединять файл MS Word, описывающий вариант использования, с описанием требования в репозитории (точнее, делает это очень неудобно, с отказом от структурирования исходного текста);

б) поддерживает интеграцию со средствами моделирования (IBM Rational XDE и Rose), но не через Web-интерфейс, что в данном случае было критически важно. Вообще оказалось, что для корректной работы репозитория через Web нужно избегать работы по локальной сети с помощью «толстого клиента» данного пакета, иначе могут появиться документы, доступ к которым возможен только из локальной сети;

в) не позволяет задать права доступа различных пользователей к отдельным требованиям (в данном случае работу с репозиторием требований параллельно вели около десяти человек);

г) требует формирования и ведения списка терминов (глоссария) проекта, списка действующих лиц (участников вариантов использования), а также реестра регламентирующих документов (документов, на которых базируется работа системы) также в репозитории требований (данные документы представлялись в пакете как специфические требования к ИС);

д) в качестве наиболее удобного варианта структурирования требований предлагает только вариант, при котором требования группируются в оглавления (пакеты) и детализируются требованиями следующего уровня (связь предок – потомок (parent – child)), что затрудняло назначение артефактов других типов;

е) позволяет организовать трассировку между артефактами (в том числе требованиями) любых типов, однако сама трассировка должна быть только одного типа;

ж) не реализует удобный механизм организации связи репозитория требований с внешними ресурсами, расположенными в Internet (с помощью URL), а также с внутренними ресурсами (с помощью гиперссылок).

В-четвертых, важную роль в процессе сбора и управления требованиями сыграло решение разработчика о периодической генерации и публикации ТЗ на ИС на основе репозитория требований. Периодическая генерация ТЗ на основе репозитория требований позволяла визуализировать то, что хранилось в трудно воспринимаемой «куче папок» самого репозитория. Таким образом, сами собранные требования стали трудно воспринимаемыми в отрыве от созданного на их основе ТЗ. При этом структура репозитория требований всё больше приближалась к структуре ТЗ, которая была предложена заказчиком.

В то же время, слишком детальная структура ТЗ, не учитывающая специфику конкретного проекта, мешала работе. Кстати, в [76] в качестве одного из упрощающих предположений, позволяющих использовать приведенную методику управления требованиями, указывалось: «Также предполагается, что не существует оговорённых контрактом требований о специальном формате документов».

В-пятых, этап согласования ТЗ с большим количеством представителей заказчика прошел достаточно успешно. Для учёта поступивших замечаний был создан такой тип требований, как «замечания». Каждое такое замечание, снабжённое именем его автора, было связано трассировкой с соответствующим требованием, по поводу которого оно было высказано (т.е. с предложением об изменении ТЗ). После этого работа по учёту замечаний в ТЗ велась параллельно несколькими членами команды: каждый мог внести изменение в требования, а потом соответственно изменить статус «замечания» (принято/учтено/отклонено…). В качестве результата такой работы Ю.О. Волков отмечает: «… в репозитории на 400 своих требований мы получили 200 замечаний» [87]. Однако набор трассировок, созданных с разными целями, постепенно превратился в «кучу, в которой уже нет наглядности». После появления большого количества замечаний и их трассировки на требования репозиторий стал выглядеть «особенно печально».

В-шестых, сравнительно мало внимания было уделено выявлению требований к ИС в целом и ее взаимодействию с другими ИС, которые уже эксплуатируются в БП заказчика. Это привело к трудностям позиционирования разрабатываемой ИС как в охватываемых автоматизацией БП заказчика, так и в ее функциональности. Кроме того, были отмечены проблемы, возникающие из-за излишней детализации требований к ИС. По словам Ю.О. Волкова: «... такими детальными требованиями просто невозможно управлять. Ведь идея состоит в том, чтобы отслеживать изменения требований, анализировать причины и следствия этих изменений и т. д. А если меняются требования на верхних уровнях, включая те уровни, которые раньше просто не были описаны, то уже некому заниматься анализом того, «какие же из детальных требований выжили». Гораздо проще, при необходимости, сформулировать детальные требования с чистого листа, т.е. на основе более общих требований, а не возиться с мусором старых требований» [87].

И, наконец, в-седьмых, использование подобного подхода к управлению требованиями для отдельной ИС, разрабатывавшейся по уникальному заказу, оставляет открытым вопрос повторного использования сформированного репозитория требований к этой ИС. Описанные выше проблемы его формирования и использования заставляют разработчиков задуматься над возможностью отказа от этого репозитория и повторного решения задачи управления требований «с самого начала» даже для модификации этой ИС.

Изложенные результаты позволяют утверждать, что высказывать претензии к непосредственным исполнителям работ по формированию и управлению требованиями не совсем справедливо. Даже соглашаясь с низким уровнем квалификации многих исполнителей, вопросы применимости излагаемых в литературе общих методов и методик формирования, анализа и управления требованиями остаются открытыми. Как видно из приведенного выше примера, многие ошибки возникают из-за невозможности использовать «в чистом виде» преимущества той или иной теоретической методики или инструментального сред-

ства в ходе проектирования конкретной ИС. Не менее важной является проблема повторного использования накопленных требований и соответствующих этим требованиям результатов разработки ИС и ее отдельных элементов в проектах других ИС. Последнее приобретает особую значимость для создания сервис-ориентированных ИС.

1.8 Выводы

На основании изложенных в разделе сведений можно сделать следующие выводы.

Во-первых, существующие определения термина «информационная система» и основные направления уточнения этих определений позволяют однозначно утверждать: основная проблема создания ИС заключается в правильной организации таких последовательностей представлений данных, которые обеспечивали бы формирование единого целостного информационного представления объекта или процесса, необходимого для достижения поставленных перед ИС целей. При этом проблема программирования операций обработки данных, образующих такие представления, является вторичной. С математической точки зрения ИС может рассматриваться как прообраз всех систем, реализованных на отдельных видах элементов комплекса средств автоматизации (видов обеспечений ИС) – например, базы данных, программной системы, системы комплексов технических средств и т.п. Иными словами, невозможно разработать, например, эффективную программную систему по заказу предприятия или организации до тех пор, пока не будет разработана эффективная ИС управления этим предприятием или организацией.

Во-вторых, основные направления развития ФС ИС заключаются, прежде всего, в стандартизации и унификации представлений отдельных функций и ФЗ ИС. Наблюдается также децентрализация ФС ИС, выражаемая, прежде всего, в переходе от ярко выраженных иерархических структур функциональных подсистем к формированию сетевых структур из отдельных ФМ и особенно из отдельных ИТ-услуг. Однако проблема выработки каких-либо теоретических предпосылок интеграции ФЗ, ФМ или ИТ-услуг в ФС ИС как единое целостное информационное представление объекта или процесса, по-прежнему не имеет достаточно хорошего коммерческого решения. Более того, решение этой проблемы по-прежнему возлагается на сотрудников подразделения, занимающегося сопровождением ИС, и, чаще всего, приводит к упомянутому выше эффекту «ИТ-слепоты».

В-третьих, основным достижением исследований в области разработки ИС и их ФС следует признать выделение и развитие типовых функций, ФЗ, ФМ и ИТ-услуг. Однако по-прежнему остаются нерешенными вопросы, связанные с существованием объективных законов формирования эффективных и качест-

венных вариантов конфигураций ФС ИС из множества типовых ФЗ или ИТ-услуг. В большинстве случаев эти вопросы решаются также на интуитивном уровне сотрудниками подразделения, занимающегося сопровождением ИС. Рассмотренный в подразделе 1.3 процессно-архитектурный подход к описанию жизненного цикла ИС в качестве основы таких законов предлагает использовать понятие «Архитектура ИС», однако предоставляет в распоряжение специалистов лишь набор концептуальных моделей формирования архитектуры ИС, а точнее – ее описаний. При этом вопрос о возможности повторного использования описаний архитектуры конкретной ИС для создания новых систем аналогичного назначения остается нерешенным. В то же время предлагаемые концептуальные модели позволяют сформулировать научно-прикладную проблему создания специализированного компонента ИС, главной задачей которого будет являться интеграция разнородных элементов ИС в соответствии с законами, закономерностями и ограничениями, заданными при описании архитектуры этой ИС.

В-четвертых, изложенная в подразделе 1.4 информация позволяет предположить, что такие понятия, как «требование» и «требование к ИС» являются категориальными понятиями, которым нельзя дать аналитически четкое определение. Такое понятие объединяет в себе целый ряд элементарных понятий, обозначающих отдельные группы требований к ИС, объединенные по какому-либо общему признаку. В то же время рассмотренные в подразделе 1.4 способы разделения требований на группы, несмотря на определенное сходство, все же в достаточной степени различаются между собой. Кроме того, как уже отмечалось выше, выделение таких признаков классификации не носит объективного научного характера и является следствием практического применения групп требований для описания того или иного аспекта проектируемой системы.

В-пятых, в большинстве материалов, посвященных проблеме описания требований при помощи множества атрибутов, указывается, что предлагаемые атрибутивные описания требований различных групп основаны на анализе опыта успешного/неуспешного выполнения конкретных проектов создания ПО, ИС или ИТ. В то же время теоретико-прикладной вопрос о необходимости и достаточности предлагаемых атрибутивных описаний требований в условиях конкретного проекта создания ИС остается нерешенным. Решение этого вопроса позволило бы выделить некие инварианты (или, на прикладном уровне, паттерны) описаний требований, которые, оставаясь неизменными для проекта любого (или почти любого) типа, позволяли бы формировать на своей основе описания требований проекта создания конкретной ИС с гарантированной возможностью их повторного использования в другом проекте.

В-шестых, современные способы организации управления требованиями являются следствиями ситуации, сложившейся в области определений и описаний понятия «требование». Большинство современных стандартов, литературных и Интернет-источников [56, 75-80, 83, 86, 87] рассматривают процессы управления требованиями как совокупность лучших практик, отобранных в ре-

зультате изучения опыта успешных проектов. Однако в изложении этих практик почти ничего не говорится об условиях и факторах, обеспечивавших успех применения этих практик. В результате, как показано в подразделе 1.7, возникает необходимость своеобразного симбиоза подобных практик, методов и методик формирования, анализа и управления требованиями, пытаясь при этом сформировать некую компромиссную методологию для разработки конкретной системы. При этом реализация такой компромиссной методологии опирается на инструментальные средства и программные продукты, созданные в рамках какой-либо одной из участвовавших в симбиозе методологий. Такое расхождение во многом ограничивает успех применения средств управления требованиями и делает затруднительным повторное использование наполненных репозиториев требований в других аналогичных проектах.

Аспект повторного использования требований приобретает особую значимость в условиях проектирования, внедрения, эксплуатации и модернизации ИС как совокупности ИТ-услуг. Эффект от повторного использования ИТ-услуг при условии минимума изменений их обеспечивающей части можно считать прямо пропорциональным степени повторного использования требований к этим ИТ-услугам, в том числе – с учетом преобразования описаний в терминах других предметных областей. Если же учесть неизбежность изменений требований, возникающих в ходе создания ИС, то проблема создания формальных представлений требований и механизмов управления ими, способных воплотиться в конкретные ИТ, управляющие процессами интеграции разнородных элементов ИС, по-прежнему является актуальной и требует решения.

2 ФОРМИРОВАНИЕ И АНАЛИЗ ТРЕБОВАНИЙ К ИНФОРМАЦИОННОЙ СИСТЕМЕ В РАМКАХ СЕРВИСНОГО ПОДХОДА

2.1 Сервисный подход к созданию информационных систем: основные определения

Как показано в разделе 1, с современной точки зрения ИС являются системами, образованными множеством взаимосвязанных и экономически целесообразных ИТ-услуг, своевременное предоставление и выполнение которых обеспечивает эффективную и качественную деятельность управляемого объекта и/или процесса. Подход, на основании которого сформировано данное представление ИС, здесь и в дальнейшем будем называть сервисным подходом.

Основным термином, определяющим главные особенности сервисного подхода, является термин «ИТ-услуга». Этот термин, как уже отмечалось в сноске подраздела 1.2, является, по сути, вариантом перевода оригинального термина «IT-service». Однако то же самое слово – «service» – используется также, например, для обозначения элементов программного обеспечения ИС или ИТ, реализующего законченную функцию предоставления или обработки данных, переводя их из одного целостного состояния в другое [88]. Появление и развитие сервисного подхода к формированию представления ИС выделило термин «ИТ-сервис» как самостоятельный, имеющий множественное толкование. Анализ изменения семантики данного термина применительно к основному способу реализации ИТ-сервисов как web-ориентированных приложений («web-сервис») позволяет выделить четыре основных этапа [89].

На первом этапе (до 2001 г.) этот термин используется, главным образом, для обозначения web-ориентированных ИС различного назначения. Примером такой трактовки может служить следующее определение: «Web-службы (иногда называются сервисами приложений) – это услуги (как правило, в том числе сочетание программ и данных, а, возможно, и человеческих ресурсов), которые доступны на web-сервере для web-пользователей или других web-программ» [90]. Сравнивая его с определением термина «автоматизированная система», принятым в действующих стандартах, увидим схожесть этих определений.

Однако уже на втором этапе (2002 – 2007 гг.) термин «web-сервис» меняет свой смысл. Теперь этим термином называют специализированные программы, обеспечивающие унификацию взаимодействия разнородных программ и средств программирования в рамках web-сайта или web-приложения. Примерами такой трактовки могут служить следующие толкования:

а) «Благодаря веб-сервисам, функции любой программы могут стать доступными через Интернет. Таким образом, такие программы как PHP, ASP, JSP скрипты, JavaBeans, COM-объекты и все остальные наши любимые средства программирования могут теперь обращаться к какой-нибудь программе, работающей на другом сервере (т.е. к веб-сервису), и использовать ответ, полученный от нее на своем веб-сайте, или приложении» [91];

б) «Веб-сервисы – это небольшие блоки кода. Веб-сервисы предназначены для работы с ограниченным набором задач. Веб-сервисы используют XML на основе протоколов связи. Веб-сервисы не зависят от операционных систем. Веб-сервисы не зависит от языков программирования. Веб-сервисы соединяют людей, системы и устройства» [92].

Третий этап развития семантики термина «web-сервис» (2008 – 2010 гг.) характеризуется использованием данного термина для обозначения основных элементов архитектуры построения информационных, а точнее – программных и программно-технических систем. Примером такой трактовки может служить следующее определение: «Веб-сервисы представляют собой распределенную компьютерную архитектуру, состоящую из множества разных компьютеров, связанных сетью в одну систему. Они состоят из набора стандартов, что позволяет разработчикам реализовать распределенные приложения с помощью радикально отличающихся инструментов, предоставляемых различными поставщиками для создания приложений, использующих комбинацию программных модулей, вызываемых системой в разных отделах или из других компаний» [93].

Настоящее время (то есть период с 2011 г.) можно охарактеризовать как четвертый этап развития, на котором возникает и утверждается многозначность семантики термина «web-сервис». Эта многозначность выражается, главным образом, в допустимости одновременного существования трактовок термина, принятых на втором и третьем этапах [94]. Кроме того, продолжает развиваться трактовка термина «web-сервис» как некоей услуги, предоставляемой пользователю специальными программными системами.

Поэтому здесь во избежание терминологической путаницы понятия «ИТ-сервис» и «ИТ-услуга» предлагается разделить [89].

Понятием ***«ИТ-сервис»*** в процессах разработки, внедрения, сопровождения и модернизации ИС следует описывать ***совокупность различных средств комплекса средств автоматизации, реализующих законченную операцию предоставления или обработки данных, переводя их из одного целостного состояния в другое, используя при этом стандартные платформо-независимые интерфейсы***.

Тогда понятие ***«ИТ-услуга»*** следует использовать для описания ***взаимосвязанной совокупности ИТ-сервисов, которая предоставляется для выполнения отдельной работы процесса предприятия/организации или для управления этой работой***.

Данные определения позволяют рассматривать ИТ-услугу как аналог ФЗ ИС, результат которой используется персоналом в ходе выполнения отдельной работы процесса предприятия/организации или в ходе управления этой работой. В то же время возможность представления процессов предприятия/организации как отдельных работ, которые, в свою очередь, могут быть разделены на более мелкие работы, позволяет трактовать ИТ-услугу и как аналог функции ИС, целью которой в общем случае является повышение эффективности и/или качества выполнения и/или управления соответствующим процессом предприятия/организации. Что же касается термина «ИТ-сервис», то его следует воспринимать как один из способов организации комплекса средств автоматизации, который обеспечивает выполнение функций и отдельных ФЗ ИС.

Такое установление соответствия позволяет по-иному сформулировать приведенное выше определение термина «ИТ-услуга». Согласно этой формулировке, ***ИТ-услуга – это самостоятельная ФЗ ИС, использование которой для выполнения отдельной работы процесса предприятия/организации или для управления этой работой экономически и технически целесообразно***. Такое определение позволяет рассматривать главную цель деятельности ИС предприятия или организации как формирование и отображение единого целостного информационного представления процессов этого предприятия или организации в результате оказания совокупности ИТ-услуг [89].

Следует обратить особое внимание на то, что использование сервисного подхода в процессах жизненного цикла ИС, рассмотренных в подразделе 1.3, не дает никаких оснований утверждать, что данная ИС будет иметь исключительно сервис-ориентированную архитектуру своего программного обеспечения. Сервисный подход, в отличие от существовавших ранее, позволяет установить соответствие экономически и технически обоснованных потребностей заинтересованных лиц в автоматизации управления объектами и/или процессами множеству конкретных ИТ-услуг и реализующих эти услуги ИТ-сервисов, позволяющих удовлетворить эти потребности.

Контекстная диаграмма классов, отражающая основные взаимосвязи базовых понятий сервисного подхода к созданию ИС как множества взаимосвязанных элементов, способных оказывать заинтересованным лицам услуги по выполнению операций над данными, необходимых для достижения главной цели деятельности ИС, приведена на рис. 2.1.

Основная часть понятий и терминов, показанных на рис. 2.1, рассмотрена в подразделе 1.3. Каждая проблема заинтересованных лиц (класс «Concern») может быть выражена множеством потребностей различной природы (класс «Need»), высказываемых заинтересованными сторонами в самой различной форме. Каждая из этих потребностей может быть удовлетворена одной или несколькими ИТ-услугами (класс «IT-accommodation») или же одним или

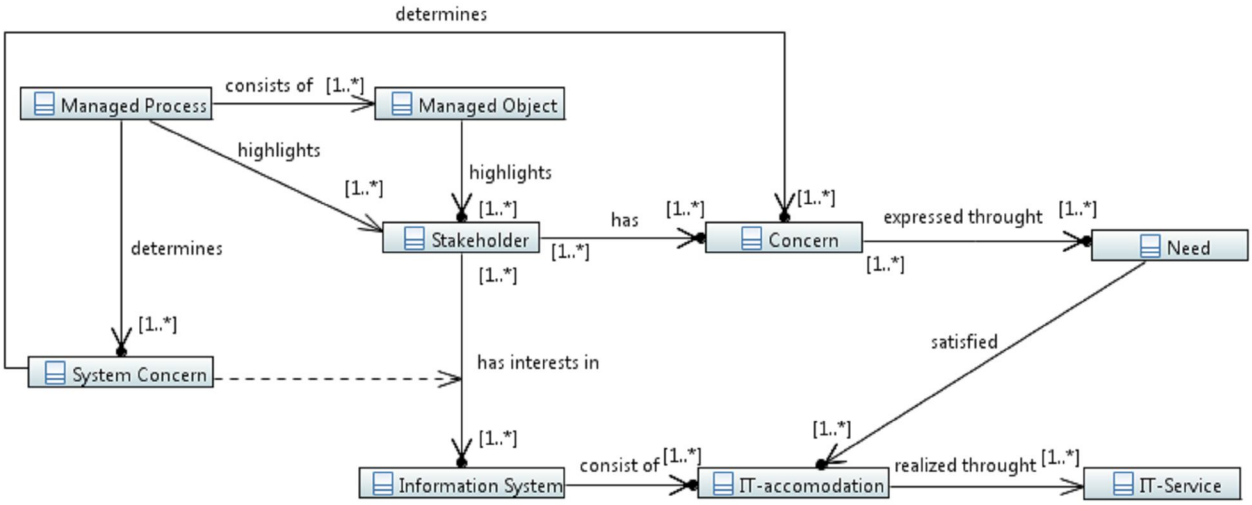

Рисунок 2.1 – Контекстная диаграмма классов понятия «сервисный подход к созданию информационных систем»

несколькими ИТ-сервисами (класс «IT-service»), реализующими соответствующие ИТ-услуги. Тогда любая ИС может быть представлена как система, состоящая из множества ИТ-услуг, а любая ИТ-услуга может быть реализована множеством ИТ-сервисов.

Подобное представление ИС позволяет установить единую терминологическую базу для описания ФС и обеспечивающей части ИС. Например, ФМ ИС с точки зрения сервисного подхода представляет собой подмножество ИТ-услуг по автоматизации выполнения неких имеющих сходные признаки функций ИС, которое может эксплуатироваться и как самостоятельная ИС, и как составная часть другой, более крупной ИС. Программное обеспечение ИС с точки зрения сервисного подхода представляет собой систему программных ИТ-сервисов, реализующих множество ИТ-услуг ИС.

Сформулированные определения понятий «ИТ-услуга» и «ИТ-сервис» позволяют выделить и уточнить основные уровни представления понятия «информационная система», определение которого приведено в подразделе 1.1. Следует отметить, что формирование подобных уровней проводилось и ранее – уже в 1970 – 1980-х гг. было выработано представление ИС как совокупности функциональной и обеспечивающей частей [11, 14, 17, 95]. Однако применение сервисного подхода к созданию ИС заставляет пересмотреть данные уровни представления. Так, например, представление ИС как системы ИТ-сервисов позволяет определить функцию ИС как законченную операцию предоставления или обработки данных. Следствием такого определения функции является рассогласование между представлениями функций ИТ-услуг и реализующих эти услуги ИТ-сервисов. Такое рассогласование становится особенно сильным в случае, если Поставщик формирует ИТ-услуги ИС из типовых ИТ-сервисов, созданных различными разработчиками. Получаемая в результате система ИТ-сервисов будет наряду с операциями предоставления или обработки

данных, реализующими необходимые ИТ-услуги, осуществлять и операции, которые не используются Потребителем в формируемой ИС. Такие неиспользуемые операции могут приводить к ухудшению значений показателей эффективности и/или качества функционирования ИС (например, к снижению уровня защищенности ИС за счет присутствия операций, обеспечивающих возможность несанкционированного доступа к данным).

Эта и другие ей подобные ситуации требуют пересмотра сложившихся уровней представления. Вместо уровней представления ИС, выделяющих функциональную и обеспечивающую части, предлагается использовать следующие уровни представления создаваемой ИС:

- уровень управляемых объектов и/или процессов;
- общесистемный уровень;
- уровень ИТ-услуг;
- уровень ИТ-сервисов.

Понятия, выделенные на уровне управляемых объектов и/или процессов, используются для описания тех особенностей предметной области, которые непосредственно влияют на систему, ее архитектуру и описания этой архитектуры. Понятия, выделенные на общесистемном уровне, используются для определения ИС как единой целостной сущности, архитектуры ИС в целом и описаний этой архитектуры. Понятия, выделенные на уровне ИТ-услуг, используются для описания отдельных ИТ-услуг и их множества, образующего ИС. Понятия, выделенные на уровне ИТ-сервисов, используются для описания отдельных ИТ-сервисов и их множеств, реализующих ИС как множество ИТ-услуг.

Уточнения определения понятия «информационная система» на каждом из предлагаемых уровней представления создаваемой ИС приведены в табл. 2.1. Указанные в скобках традиционные уровни представления позволяют соотнести предлагаемые уточнения со сложившимися определениями функциональной и обеспечивающей части ИС.

В соответствии с уточнениями понятия «информационная система», приведенными в табл. 2.1, можно уточнить понятия «архитектура» и «описание архитектуры» в рамках сервисного подхода к созданию ИС. Следует отметить, что в настоящее время понятие «архитектура» может использоваться для описания фундаментальных понятий и свойств как ИС, так и функций ИС или же отдельных видов обеспечений ИС. В результате возникает терминологическая путаница, которая может привести к неоправданно узкому взгляду на разрабатываемую ИС (например, представление ИС как системы, архитектура которой определяется исключительно особенностями программного обеспечения этой системы). Кроме того, выделение и уточнение понятий «архитектура» и «описание архитектуры» на уровнях ИТ-услуг и ИТ-сервисов может позволить выделить и, возможно, формализовать основные принципы проектирования и развития комплекса ИТ-услуг и ИТ-сервисов ИС.

Таблица 2.1 – Определения понятия «информационная система» на разных уровнях представления

Уровень представления	Определение понятия «информационная система»
Уровень управляемых объектов и/или процессов	ИС – один из механизмов управляемого объекта и/или процесса, формирующий и отображающий единое целостное информационное представление этого объекта и/или процесса в соответствии с поставленными перед этим механизмом целями.
Общесистемный уровень	ИС – система, состоящая из персонала и комплекса средств автоматизации и направленная на формирование и отображение единого целостного информационного представления объекта или процесса в соответствии с поставленными перед ней целями.
Уровень ИТ-услуг (функциональная часть ИС)	ИС – система ИТ-услуг, использование которой для объекта или процесса экономически или технически целесообразно, направленная на формирование и отображение единого целостного информационного представления этого объекта или процесса в соответствии с поставленными перед ней целями.
Уровень ИТ-сервисов (обеспечивающая часть ИС)	ИС – система ИТ-сервисов, реализующих совокупность операций предоставления или обработки данных, направленная на формирование и отображение единого целостного информационного представления объекта или процесса в соответствии с поставленными перед системой целями.

Контекстная диаграмма классов понятия «описание архитектуры информационной системы на основе сервисного подхода» приведена на рис. 2.2.

Основываясь на рассмотренном в подразделе 1.3 определении понятия «архитектура системы», понятие «архитектура ИТ-услуг» можно определить следующим образом: ***«Архитектура ИТ-услуг – фундаментальные понятия и свойства комплекса ИТ-услуг, образующих ИС в окружающей её среде, воплощенные в элементах и отношениях этого комплекса, а также в принципах его проектирования и развития»***. Примером понятий, образующих архитектуру ИТ-услуг, может служить понятие «функция ИС», которое устанавливает представление ИТ-услуги как преобразования множества входных данных в множество выходных данных. Примером принципов проектирования комплекса ИТ-услуг, в которых воплощается

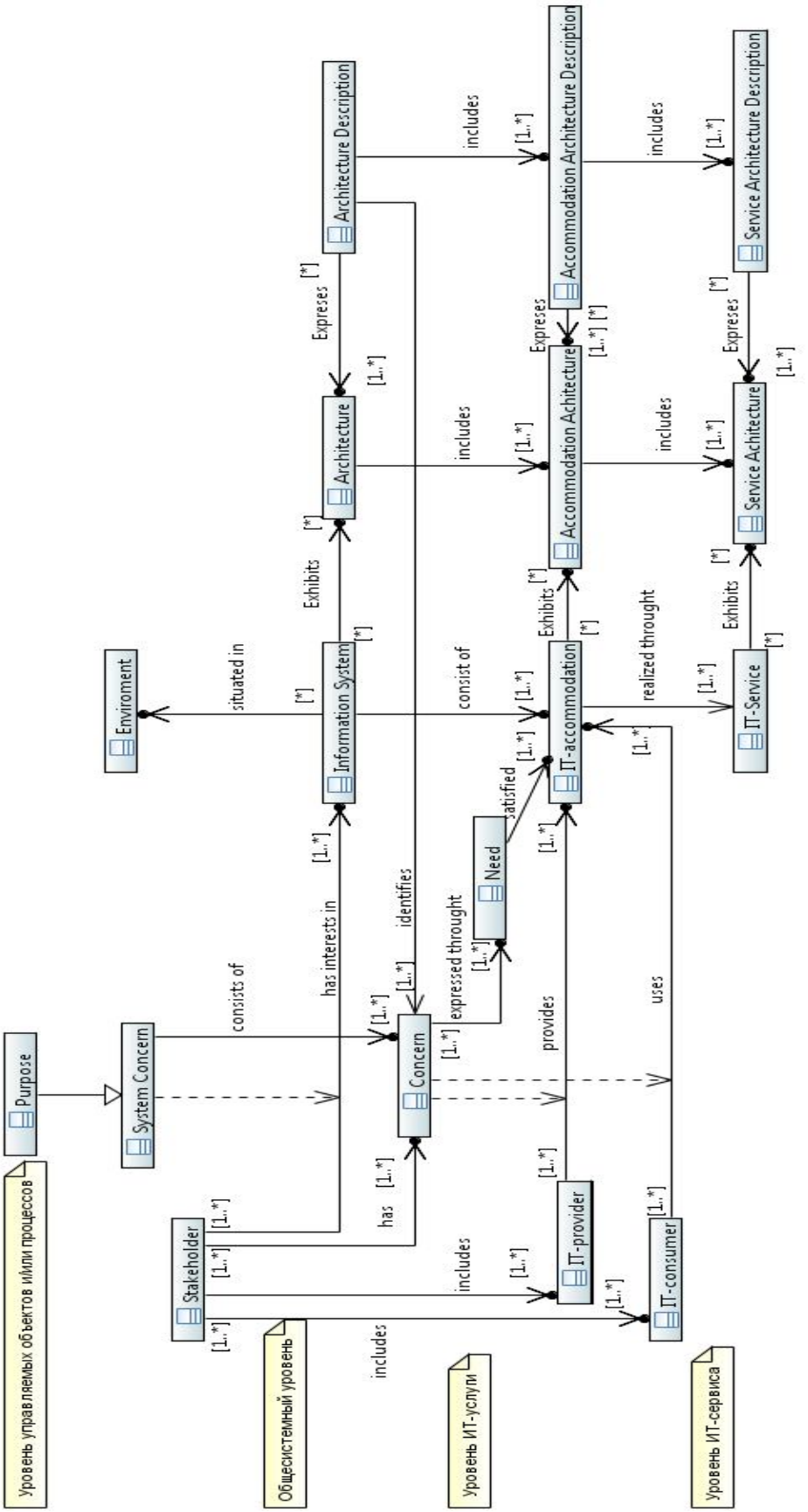

Рисунок 2.2 – Контекстная диаграмма классов понятия «описание архитектуры системы на основе сервисного подхода»

архитектура ИТ-услуг, может служить принцип модульности ИТ-услуг, устанавливающий возможность одновременного представления некоего подмножества ИТ-услуг и как самостоятельно функционирующей ИС, и как элемента более крупной ИС.

Тогда понятие «архитектура ИТ-сервисов» можно определить следующим образом: ***«Архитектура ИТ-сервисов – фундаментальные понятия и свойства комплекса ИТ-сервисов и связывающих их интерфейсов, образующих ИС в окружающей её среде, воплощенные в элементах и отношениях этого комплекса, а также в принципах его проектирования и развития»***. Примером понятий, образующих архитектуру ИТ-сервисов, могут служить понятия «клиент» и «сервер», которые устанавливают необходимость разделения программных сервисов на отдельные элементы, которые должны размещаться и функционировать на удаленных друг от друга комплексах технических средств.

Не менее важным уточнением является уточнение понятия «заинтересованные стороны»[9], которое на уровне управляемых объектов и/или процессов обозначает всех участников работ по созданию, внедрению, эксплуатации и модернизации ИС как совокупности ИТ-услуг. В настоящее время наиболее полный и точный перечень таких участников приведен в стандарте ГОСТ 34.601-90. Согласно этому перечню, в процессах создания, внедрения, эксплуатации и модернизации АС могут принимать участие следующие стороны [96]:

а) организация-заказчик (пользователь) – организация, для которой создаются АС и которая обеспечивает финансирование, приемку работ и эксплуатацию АС, а также выполнение отдельных работ по созданию АС;

б) организация-разработчик – организация, которая осуществляет работы по созданию АС, представляет заказчику совокупность научно-технических услуг на разных стадиях и этапах создания, а также разрабатывает и представляет различные программные и технические средства АС;

в) организация-поставщик – организация, которая изготавливает и поставляет программные и технические средства по заказу разработчика или заказчика;

г) организация-генпроектировщик объекта автоматизации;

д) организации-проектировщики различных частей проекта объекта автоматизации для проведения строительных, электротехнических, санитарно-технических и других подготовительных работ, связанных с созданием АС;

е) организации строительные, монтажные, наладочные и другие.

Данный перечень не является жестким разграничением ролей участников и допускает в зависимости от условий создания АС возможность различных

[9] В [50] термин «заинтересованная сторона» переведен как «правообладатель» и трактуется следующим образом: «Сторона, имеющая право, долю или претензии на систему или на владение ее характеристиками, удовлетворяющими потребности и ожидания этой стороны».

совмещений функций заказчика, разработчика, поставщика и других организаций, участвующих в работах по созданию АС [96].

Однако предлагаемый ГОСТом 34.601-90 перечень участников ориентирован на вполне определенное представление ИС, сформировавшееся в 1980-х гг. Представление ИС как совокупности ИТ-услуг заставляет пересмотреть основные подходы к выделению ролей участников. С этой целью рассмотрим современные зарубежные стандарты, посвященные проблемам создании и эксплуатации ИС.

Одним из таких стандартов является международный стандарт ISO/IEC 20000. Это международный процессно-ориентированный стандарт управления ИТ-услугами. Цель его разработки заключалась в создании универсальных критериев, с помощью которых любая фирма или служба, предоставляющая ИТ-услуги, сможет оценивать их эффективность и выполнение требований заказчиков с учетом их бизнеса [97]. В стандарте ISO/IEC 20000 основным понятием, определяющим участников процессов оказания ИТ-услуг, является понятие «поставщик услуг». Это понятие определяет организацию, которая поставила перед собой цель соответствовать требованиям ICO/IEC 20000. В некоторых случаях поставщиком услуг может являться внутреннее подразделение организации, например, подразделение, занимающееся внедрением и эксплуатацией ИТ в организации [98].

Следует отметить, что стандарт ICO/IEC 20000 делает основной упор в определении понятия «поставщик услуг» на возможность представления соответствующей организации, прежде всего, как самостоятельного предприятия, внешнего по отношению ко всем остальным участникам процессов создания, внедрения, эксплуатации и модернизации ИТ-услуг. Представление организации-поставщика услуг как внутреннего подразделения с точки зрения стандарта ICO/IEC 20000 является частным случаем.

Однако такое определение требует дополнительного уточнения ролей, в которых может выступать поставщик ИТ-услуг в процессах создания, внедрения, эксплуатации и модернизации ИС. Так, согласно этому определению, поставщиком ИТ-услуг может быть как организация, занимающаяся разработкой и внедрением ИТ-сервисов, так и организация или подразделение, занимающееся предоставлением конечным пользователям ИТ-услуг или ИТ-сервисов, реализующих эти услуги. С точки зрения стандарта ICO/IEC 20000, особых различий здесь нет. В то же время практика создания и эксплуатации ИС различного назначения показывает, что права на отдельные виды обеспечений ИС у различных участников процессов создания, внедрения, эксплуатации и модернизации ИТ-услуг могут серьезно различаться.

Поэтому предлагается в общем случае понятием ***«поставщик ИТ-услуг»*** описывать ***организации и/или подразделения, которые основной целью своей деятельности считают эффективное и качественное выполнение работ по***

созданию, внедрению, эксплуатации и модернизации ИС, отдельных ИТ-услуг и реализующих эти услуги ИТ-сервисов.

В связи с этим целесообразно уточнить понятие «поставщик ИТ-услуг» для основных процессов жизненного цикла ИС. Для такого уточнения лучше всего использовать рассмотренный выше перечень участников. На основе этого перечня в разных процессах жизненного цикла ИС организации или подразделения-поставщики ИТ-услуг могут выступать в следующих ролях [89]:

а) разработчик – осуществляет работы по созданию ИТ-услуг, представляет заказчику совокупность научно-технических услуг на разных стадиях и этапах создания, а также разрабатывает и поставляет различные ИТ-сервисы и их элементы, обеспечивающие реализацию соответствующих ИТ-услуг;

б) поставщик – изготавливает и поставляет отдельные элементы ИТ-сервисов, обеспечивающих реализацию соответствующих ИТ-услуг, по заказу разработчика или заказчика;

в) дилер разработчика – занимается адаптацией предлагаемых разработчиком ИТ-услуг и реализующих эти услуги ИТ-сервисов к особенностям страны, региона и т.п.;

г) организация по внедрению ИТ-услуг – осуществляет работы по адаптации предлагаемых разработчиком ИТ-услуг и реализующих эти услуги ИТ-сервисов к особенностям конкретного заказчика и внедрению адаптированных ИТ-услуг и ИТ-сервисов в процессы заказчика;

д) ИТ-подразделение – осуществляет работы по непосредственному обслуживанию, настройке и администрированию ИТ-услуг и ИТ-сервисов, эксплуатируемых в процессах заказчика, работы по управлению этими ИТ-услугами и ИТ-сервисами, работы по выявлению отклонений фактических эксплуатационных характеристик ИТ-услуг от проектных значений, а также осуществление взаимодействия между заказчиком и другими участниками процессов создания, внедрения, эксплуатации и модернизации ИТ-услуг;

е) организация по сопровождению ИТ-услуг и ИТ-сервисов – осуществляет работы по сопровождению отдельных ИТ-услуг и реализующих их ИТ-сервисов, установлению причин возникновения отклонений фактических эксплуатационных характеристик ИТ-услуг от проектных значений, устранению выявленных причин и недостатков ИТ-услуг и ИТ-сервисов, а также работы по внесению необходимых изменений в документацию на ИТ-услуги и ИТ-сервисы.

Что касается понятия «потребитель ИТ-услуг», то оно в явном виде в стандарте ICO/IEC 20000 не определено. В общем случае понятием ***«потребитель ИТ-услуг и реализующих эти услуги ИТ-сервисов»*** следует описывать ***организации и/или подразделения, которые нуждаются в этих услугах и сервисах и используют их в своих процессах для достижения целей, поставленных перед этими организациями и/или подразделениями.***

Уточним понятие «потребитель ИТ-услуг» для процессов жизненного цикла ИС, аналогично рассмотренному ранее уточнению понятия «поставщик ИТ-услуг». Тогда в разных процессах жизненного цикла ИС организации или подразделения-потребители ИТ-услуг могут выступать в следующих ролях [89]:

а) заказчик ИС или отдельных ИТ-услуг – определяет основные требования к разрабатываемой ИС или же отдельным ИТ-услугам и обеспечивает финансирование работ по созданию, внедрению, эксплуатации и модернизации ИС и соответствующих ИТ-услуг, а также выполнение отдельных работ по созданию, внедрению и модернизации ИС или отдельных ИТ-услуг;

б) пользователь ИТ-услуг и реализующих эти услуги ИТ-сервисов – обеспечивает приемку работ по созданию и эксплуатации ИТ-услуг (в том числе в рамках ИС) и реализующих эти услуги ИТ-сервисов, а также выполнение отдельных работ по созданию, внедрению и модернизации ИТ-услуг и реализующих эти услуги ИТ-сервисов.

Сформулированные выше определения основных понятий сервисного подхода к созданию ИС позволяют установить цели участников, а также основные особенности процессов проектирования, внедрения, эксплуатации и модернизации ИС как совокупности взаимосвязанных и экономически целесообразных ИТ-услуг, своевременное предоставление и выполнение которых обеспечивает эффективную и качественную деятельность управляемого объекта и/или процесса.

2.2 Анализ основных процессов, работающих с требованиями к информационной системе

Рассмотренные в подразделе 2.1 основные положения сервисного подхода к созданию ИС определяют необходимость анализа существующих описаний процессов жизненного цикла ИС, работающих с требованиями к ИС. Как отмечалось в подразделе 1.2, в общем случае под процессом следует понимать совокупность взаимосвязанных и взаимодействующих видов деятельности, преобразующих входы (материальные, информационные потоки и т.п.) в выходы, представляющие ценность для потребителя. В соответствии с этим определением для описания процесса необходимо установить значения следующих атрибутов (согласно стандарту ISO/IEC 24774:2007):

- название (наименование) процесса;
- выходы процесса;
- виды деятельности, выполняемые в рамках процесса.

Процессы, работающие с требованиями к ИС, в стандарте ISO/IEC 15288:2002 относятся к группе технических процессов (см. подраздел 1.3). Среди технических

процессов непосредственно с требованиями к ИС работают два следующих процесса: процесс определения требований правообладателей и процесс анализа требований. Результаты этих процессов являются входами для процесса проектирования архитектуры системы, целью которого является синтез решения, удовлетворяющего системным требованиям [50].

Процесс определения требований правообладателей позволяет определить правообладателей или классы правообладателей, которые связаны с системой на протяжении всего жизненного цикла, а также их потребности и пожелания. В рамках процесса эти данные анализируются и преобразуются в общий набор требований правообладателей, описывающих ожидаемое поведение системы в процессе взаимодействия с эксплуатационной средой, и совокупность базовых показателей, проверка на соответствие которым является целью процесса валидации, позволяющего подтвердить, что система отвечает заявленным требованиям [50].

В ходе процесса анализа требований создается представление о будущей системе, которая сможет удовлетворить требования правообладателей и, если позволят ограничения, не подразумевает какой-либо специфической реализации. В результате данного процесса задаются измеримые системные требования, зависящие от видения разработчика, в которых определяется, какими характеристиками должна обладать система и какими должны быть значения этих характеристик, чтобы удовлетворить требования правообладателей [50].

Описания процессов определения требований правообладателей и анализа системных требований приведены в табл. А.1 Приложения А.

Для программных систем положения стандарта ISO/IEC 15288:2002 могут быть уточнены стандартом ISO/IEC 12207:2008, который был принят в России в 2010 г. Данный стандарт предназначен для представления определенной совокупности процессов, облегчающих связи между приобретающими сторонами, поставщиками и другими правообладателями в течение жизненного цикла программных продуктов [99].

В соответствии с положениями стандарта ISO/IEC 12207:2008 процессы, непосредственно работающие с требованиями, относятся к следующим группам процессов:

а) технические процессы;

б) процессы жизненного цикла программных средств.

Данные группы процессов выделены для обеспечения взаимосвязи со стандартом ISO/IEC 15288:2002. Группа технических процессов определяет системный контекст для работы с автономным программным продуктом или услугой, или программной системой. Группа процессов жизненного цикла программных средств содержит специальные процессы программных средств для использования в реализации программного продукта или услуги, которые являются некоторым элементом более крупной системы [99].

В группе технических процессов стандарт ISO/IEC 12207:2008 выделяет два основных процесса, непосредственно работающих с требованиями к программным системам – процесс определения требований правообладателей и процесс анализа системных требований. При этом в стандарте отмечается, что указанные процессы являются специальными случаями аналогичных процессов стандарта ISO/IEC 15288:2002: «Пользователи могут рассматривать требуемое соответствие по отношению к процессу в стандарте ISO/IEC 15288:2002 в большей степени, чем к процессу в настоящем стандарте» [99].

Описания процессов определения требований правообладателей и анализа системных требований приведены в табл. А.2 Приложения А.

В группе процессов жизненного цикла программных средств выделяется один процесс, непосредственно работающий с требованиями, – процесс анализа требований к программным средствам. С точки зрения стандарта ISO/IEC 15288:2002, данный процесс предусматривается процессом анализа требований стандарта ISO/IEC 15288:2002 при рекурсивном применении этого процесса [99].

Описание процесса анализа требований к программным средствам приведено в табл. А.3 Приложения А.

Анализ описаний рассмотренных процессов показывает следующее.

Во-первых, в стандарте ISO/IEC 15288:2002 присутствует лишь косвенное разделение требований на функциональные и нефункциональные, а требования, подобные бизнес-требованиям (см. подраздел 1.4), отсутствуют полностью. Это означает, что данный стандарт не рассматривает зависимости функциональных и нефункциональных требований к системе от бизнес-требований, обусловленных экономико-техническими особенностями объекта, как обязательные.

Во-вторых, в стандарте ISO/IEC 12207:2008 присутствует выделение трех групп требований, описывающих разрабатываемую программную систему на трех различных уровнях представления. Однако эти группы требований не вполне соответствуют группам, рассматриваемым в большинстве литературных источников (см. подраздел 1.4). С другой стороны, стандарт ISO/IEC 12207:2008 опубликован позже, чем большинство соответствующих литературных источников и, казалось бы, должен учесть накопленный опыт формирования, обработки и анализа требований к программным системам. Однако и здесь возникает проблема несоответствия определений групп требований. Пример такого несоответствия приведен в табл. 2.2.

В то же время следует учесть, что описания процессов, приведенные в стандартах ISO/IEC 15288:2002 и 12207:2008, не представляют конкретного подхода к осуществлению этих процессов, как и не предопределяется модель жизненного цикла системы или программного средства, методология или технология их разработки. Вместо этого данные описания предназначаются для

Таблица 2.2 – Соотношение групп требований согласно стандарту ISO/IEC 12207:2008, определению К.И. Виггерса и определению Д. Леффингуэлла и Д. Уидрига

Наименование группы требований согласно ISO/IEC 12207:2008	Наименование группы требований согласно Карлу И. Виггерсу ([77])	Наименование группы требований согласно Д. Леффингуэллу и Д. Уидригу ([76])
Требования правообладателей	Бизнес-требования	Область проблемы (технические или бизнес-задачи предметной области)
	Требования пользователей, описывающие цели и задачи, которые пользователям позволит решить система	Потребности заинтересованных лиц, в том числе пользователей системы
	Системные требования	-
Системные технические (функциональные и нефункциональные) требования	Функциональные требования, определяющие функциональность ПО	Функции системы
	Нефункциональные требования, описывающие цели и атрибуты качества функций системы	Ограничения проектирования
Требования к программным элементам системы	-	Функциональные требования к программному обеспечению
	-	Нефункциональные требования к программному обеспечению

принятия организацией и базируются на деловых потребностях организации и области приложений. Определенный организацией процесс принимается в проектах организации в контексте требований заказчиков [99]. Иными словами, приведенные в стандартах ISO/IEC 15288:2002 и 12207:2008 описания являются своеобразными паттернами, в которых указывается, что следует делать, но не говорится, как именно это следует делать. Для использования этих процессов при разработке ИС Поставщик и Потребитель должны детализировать их применением конкретного архитектурного фреймворка. Выбранный Поставщиком и Потребителем архитектурный фреймворк, в свою очередь, определяет используемые сторонами в ходе разработки конкретные модели и методы.

Рассмотрим особенности процессов, непосредственно работающих с требованиями к ИС, на уровне архитектурного фреймворка. В качестве такого фреймворка выберем британскую методологию SSADM, которая является классическим примером действий по созданию ИС (преимущественно, крупномасштабных). В этой методологии уже в 1980-1990-х гг. работы по созданию ИС группировались в пять основных процессов (модулей) [73, 100]:

а) «Анализ реализуемости» (Feasibility Study, FS);

б) «Анализ требований» (Requirements Analysis, RA);

в) «Спецификация требований» (Requirements Specification, RS);

г) «Спецификация логической структуры» (Logical System Specification, LS);

д) «Физическое проектирование» (Physical Design, PD).

В настоящее время методология SSADM рассматривает процессы проектирования ИС как последовательность таких стадий [5, 101]:

- необязательная стадия 0 «Осуществимость» (Feasibility);
- стадия 1 «Исследование текущей ситуации» (Investigation of Current Environment);
- стадия 2 «Выбор бизнес-системы» (Business System Option);
- стадия 3 «Определение требований» (Definition of Requirements);
- стадия 4 «Выбор технической системы» (Technical System Option);
- стадия 5 «Логическое проектирование» (Logical Design);
- стадия 6 «Физическое проектирование» (Physical Design).

Непосредственная работа с требованиями к ИС в рамках методологии SSADM в настоящее время осуществляется на стадиях с 0 по 3 включительно. В соответствии с этим рассмотрим основные описания каждой из указанных стадий. Следует помнить, что последовательность выполнения стадий SSADM определяет выходы каждой стадии как входы последующей стадии.

Описания стадий методологии SSADM, непосредственно работающих с требованиями, приведены в табл. А.4 Приложения А.

Несколько особняком в ряду процессов работы с требованиями стоит стадия 4 SSADM. Данная стадия может выполняться параллельно стадиям 3 или 5 и заключается в разработке нескольких вариантов технической системы и последующем выборе из этих вариантов наилучшего. При этом оценивание вариантов производится с точки зрения их производительности, стоимости и влияния на организацию. Стоит отметить, что понятие «техническая система» в данном случае является своего рода аналогом термина «обеспечивающая часть ИС».

Что касается стадий 5 и 6 SSADM, при их выполнении предполагается неизменность требований, выдвинутых к ИС.

Анализ моделей стадий методологии SSADM позволяет сделать следующие выводы.

Во-первых, методология SSADM, в отличие от стандартов ISO/IEC 15288:2002 и 12207:2008, не рассматривает процессы управления проектом (более высокого уровня, чем технические процессы создания

системы) и процессы создания программных средств (более низкого уровня, чем технические процессы создания системы). Это подтверждает соотношение групп требований стандарта ISO/IEC 12207:2008 и SSADM (табл. 2.3).

Таблица 2.3 – Соотношение групп требований согласно стандарту ISO/IEC 12207:2008 и согласно стадиям методологии SSADM

Наименование группы требований согласно ISO/IEC 12207:2008	Наименование группы требований согласно SSADM
Требования правообладателей	Бизнес-требования организации
	Требования к аудиту, контролю и безопасности
Системные технические (функциональные и нефункциональные) требования	Функциональные требования к системе (требования к данным и процессам их обработки)
	Нефункциональные требования к системе (в том числе требования к аудиту, контролю и безопасности)
Требования к программным элементам системы	Отсутствуют в явном виде (спецификации требований к системе)

Во-вторых, в методологии SSADM отсутствует в явном виде группа требований к ИС в целом (или группа системных требований). Можно утверждать, что данная методология рассматривает ИС как один из вспомогательных БП, улучшающих эффективность или качество выполнения основных БП предприятия.

В-третьих, методология SSADM, как и ожидалось, более конкретна в описаниях процессов, чем стандарты ISO/IEC 15288:2002 и 12207:2008. Эта конкретность выражается, в частности, в создании и поддержке в рамках методологии SSADM четкой последовательности групп представлений: «пользователи системы – требования к системе – описания требований и пользователей в соответствующих каталогах – структурные модели системы в соответствии с выдвинутыми требованиями и описанными (и смоделированными) действиями пользователей». Однако на уровне методологии проблема преобразования одной группы в другую из этой последовательности с сохранением прямо или косвенно выявленных особенностей разрабатываемой системы не решена. Возможно, данное решение отдается создателями методологии SSADM на откуп конкретным инструментальным средствам моделирования системы, однако такой подход весьма и весьма спорен.

В-четвертых, современное представление функциональных требований в методологии SSADM приближается к модели процесса, определенной стандартом ISO/IEC 24774:2007. Такое приближение, задекларированное еще в

1993 г. [102], хоть и не является новым, но позволяет по-новому рассмотреть следующий за процессами, непосредственно работающими с требованиями, процесс проектирования архитектуры системы (согласно стандартам ISO/IEC 15288:2002 и 12207:2008). В самой методологии SSADM такая задача хоть и не поставлена явно в описаниях соответствующих стадий, однако косвенно должна решаться в ходе выбора лучшего варианта бизнес-системы.

В связи с этим целесообразно рассмотреть процессы проектирования архитектуры как на уровне системы, так и на уровне программных средств. Согласно стандарту ISO/IEC 15288:2002, данный процесс выделяет и устанавливает области решения, представленные в виде набора различных проблем управленческого, концептуального и, наконец, реализационного характера. В рамках процесса определяются и исследуются одна или несколько стратегий реализации системы со степенью детализации, соответствующей техническим и коммерческим требованиям и рискам. Исходя из этого, выбирается решение о проектировании архитектуры. Оно определяется на основе требований к набору системных элементов, из которых компонуется система. Конкретные требования, формируемые в результате этого процесса, являются основой для проведения верификации реализованной системы и для разработки стратегий комплексирования и верификации [50].

Описание процесса проектирования архитектуры, согласно стандарту ISO/IEC 15288:2002, приведено в табл. А.5 Приложения А.

Согласно стандарту ISO/IEC 12207:2008, в группе технических процессов выделяется процесс проектирования архитектуры системы, а в группе процессов жизненного цикла программных средств выделяется процесс проектирования архитектуры программных средств [99]. Данные процессы также являются специальными случаями рассмотренного выше процесса стандарта ISO/IEC 15288:2002.

Описания процесса проектирования архитектуры системы и процесса проектирования архитектуры программных средств, согласно стандарту ISO/IEC 12207:2008, приведено в табл. А.6 Приложения А.

Анализ приведенных в табл. А.5 и А.6 описаний показывает, что рассмотренные в данном подразделе последовательности процессов, непосредственно работающих с требованиями и завершающие эти последовательности процессы проектирования архитектуры ИС (или ее программной реализации), являются одним из наиболее серьезных «узких мест» проекта создания ИС. Итеративность этих процессов, отмеченная в [50, 99], свидетельствует о неоправданно больших затратах времени на их выполнение, что приводит к замедлению всего проекта. Особенно сильное отрицательное влияние подобной итеративности выполнения скажется в наиболее распространенной для большинства Поставщиков ИТ-услуг ситуации, когда требования Потребителя лишь частично перекрываются разработанными ранее решениями соответствующих ИТ-услуг. Следует также отметить, что

методология SSADM практически не рассматривает подобную ситуацию, что отрицательно сказывается на ее применении при разработке конкретных ИС.

В связи с этим возникает вопрос: возможно ли создание такого набора знаний, правил, моделей, методов или других способов регламентации выполнения рассмотренных последовательностей процессов, который был бы инвариантен по отношению к различным видам ИС? Иными словами: возможно ли создание объективной научной последовательности преобразований представлений «Потребитель ИТ-услуг – требование Потребителя – формальное представление требования Потребителя – модели ИС или ее компонентов» максимальной степени общности?

2.3 Требования к информационной системе: определение и классификация

Материалы, изложенные в предыдущих подразделах, заставляют по-новому взглянуть на определение понятия «требование к ИС». В соответствии с сервисным подходом к созданию ИС, требования к ИС, функции ИС, соответствующие этим требованиям, а также решения по видам обеспечений ИС, реализующие эти функции, являются откликами на высказанные заинтересованными сторонами проблемы, порожденные предметной областью, и потребности в конкретных возможностях ИС, которая должна решить эти проблемы. Именно эти отклики формируют множество проектных решений, образующих проект создаваемой ИС. Таким образом, сервисный подход к созданию ИС формирует представление требований к ИС как первоначальных проектных решений, определяющих все последующие проектные решения ИС.

Сказанное выше приводит к необходимости уточнения определения понятия «требование к ИС» и системы классификации этих требований. Поскольку, как показано в подразделе 1.4, самостоятельно существующего определения понятия «требование к ИС» не существует, в качестве базового определения будем использовать определение понятия «требование», сформулированное в стандарте IEEE 610.12-1990. Данное определение является наиболее общим, пригодным практически для любой отрасли компьютерных наук.

Основываясь на данном определении, понятие ***«требование к ИС»*** можно сформулировать следующим образом [103]:

а) условие или возможность, необходимые Потребителю ИТ-услуг для решения проблемы или достижения цели;

б) условие или возможность, которой должна обладать ИС или компонент ИС (ИТ-услуга, ИТ-сервис) с точки зрения Поставщика или Потребителя ИТ-услуг, соответствующие договору, стандарту, спецификации или другому официальному документу;

в) документированное представление условия или возможности, подобных описанным в первых двух определениях.

Предлагаемое определение является общим определением всех возможных групп требований к ИС, отдельным ИТ-услугам и отдельным ИТ-сервисам, реализующим эти ИТ-услуги. Для использования в конкретных проектах создания ИС это определение нуждается в уточнениях и дополнениях, для чего предлагается ввести иерархию следующих групп требований [103]:

а) бизнес-требование – условие или возможность, необходимые Потребителю ИТ-услуг для достижения своих бизнес-целей в результате выполнения соответствующих БП;

б) требование к ИС как аспекту бизнеса – условие или возможность, необходимые Потребителю ИТ-услуг для автоматизированного выполнения БП в соответствии с поставленными бизнес-целями;

в) требование к ИС в целом – условие или возможность, которой должна обладать ИС с точки зрения Поставщика или Потребителя ИТ-услуг, соответствующие договору, стандарту, спецификации или другому официальному документу;

г) функциональное требование к ИТ-услуге – возможность, которой должна обладать ИТ-услуга с точки зрения Поставщика или Потребителя ИТ-услуг, соответствующая договору, стандарту, спецификации или другому официальному документу;

д) нефункциональное требование к ИТ-услуге – условие, которому должна отвечать ИТ-услуга с точки зрения Поставщика или Потребителя ИТ-услуг, соответствующее договору, стандарту, спецификации или другому официальному документу;

е) функциональное требование к ИТ-сервису – возможность, которой должен обладать ИТ-сервис с точки зрения Поставщика или Потребителя ИТ-услуг, соответствующая договору, стандарту, спецификации или другому официальному документу;

ж) нефункциональное требование к ИТ-сервису – условие, которому должен отвечать ИТ-сервис с точки зрения Поставщика или Потребителя ИТ-услуг, соответствующее договору, стандарту, спецификации или другому официальному документу.

Схема иерархии данных групп требований приведена на рис. 2.3.

Данные группы соотносятся с группами требований к программному обеспечению, предложенными, например, К.И. Виггерсом, следующим образом (см. табл. 2.4) [103].

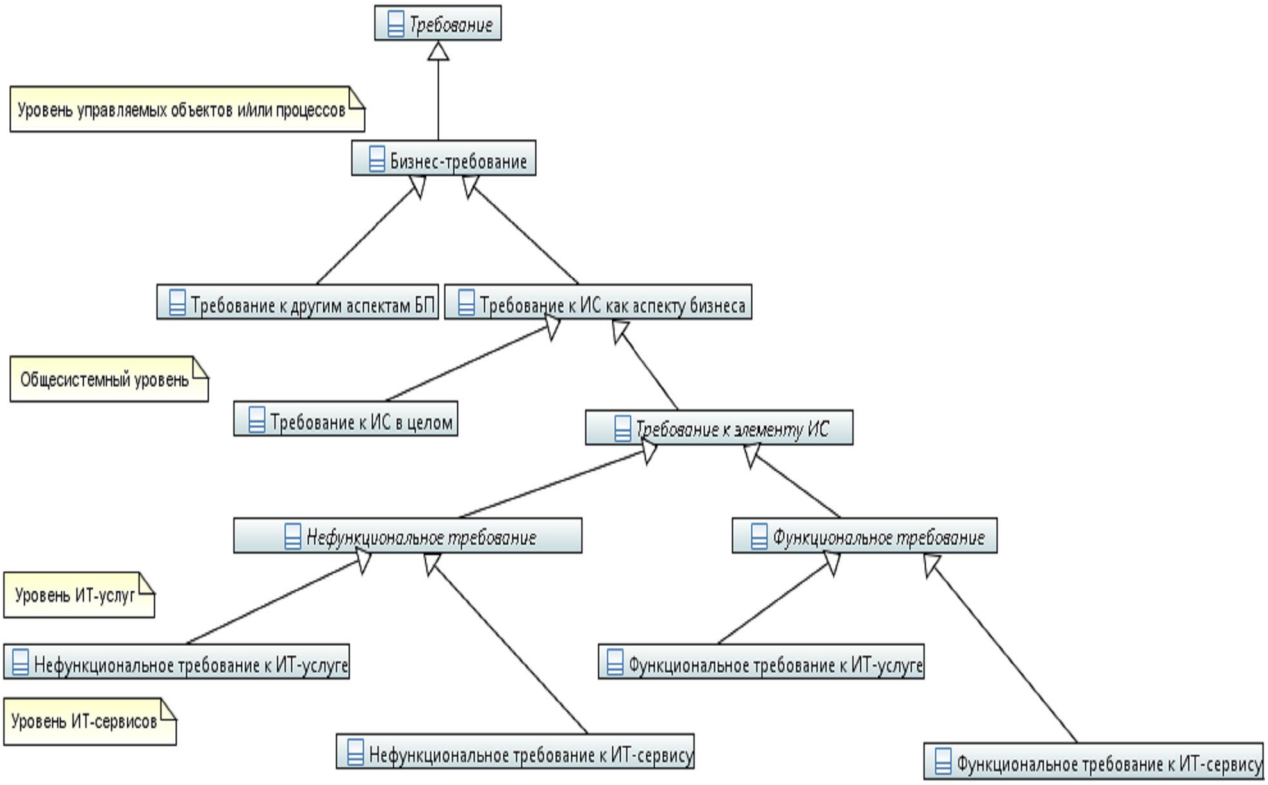

Рисунок 2.3 – Схема иерархии предлагаемых групп требований к информационной системе

Таблица 2.4 – Взаимосвязь групп требований к программному обеспечению по К.И. Виггерсу и групп требований к информационной системе

Группа требований к программному обеспечению по К.И. Виггерсу	Предлагаемая группа требований к информационной системе
Бизнес-требования	Бизнес-требования
	Требования к ИС (как к одному из аспектов бизнес-процесса)
Требования пользователей, описывающие цели и задачи, которые пользователям позволит решить система	Функциональные требования к ИТ-услугам
	Нефункциональные требования к ИТ-услугам
Функциональные требования	Функциональные требования к ИТ-сервисам
	Нефункциональные требования к ИТ-сервисам
Системные требования	Требования к ИС в целом

Предлагаемое соотношение показывает, что ИС, с точки зрения программной инженерии, представляет собой прикладное программное обеспечение, которое

предназначено для формирования и отображения единого целостного информационного представления управляемого объекта или процесса.

Таким образом, предлагаемые определение понятия «требование к ИС» и классификация групп требований соответствуют принятому в международных стандартах определению понятия «требование» и сложившимся представлениям групп требований к программному обеспечению как к более общему классу систем и определяют особенности представления ИС как совокупности ИТ-услуг, реализуемых соответствующими ИТ-сервисами.

2.4. Цели Поставщика и Потребителя ИТ-услуг и их взаимодействие в процессе проектирования архитектуры информационной системы

Предложенная в подразделе 2.3 классификация групп требований к ИС позволяет сформулировать цели Поставщика и Потребителя ИТ-услуг (далее – Поставщика и Потребителя) в ходе выполнения последовательности процессов непосредственной работы с требованиями и завершающего эту последовательность процесса проектирования архитектуры на различных уровнях представления ИС. Основное внимание следует уделить уровню представления ИС как совокупности ИТ-услуг, поскольку, как показано в подразделе 2.2, именно здесь принимаются основные архитектурные решения, определяющие облик ИС в целом.

В общем случае Поставщик и Потребитель в процессе выбора варианта конфигурации этих услуг в рамках ИС взаимодействуют следующим образом. Поставщик предоставляет Потребителю информацию о возможных вариантах конфигурации ИТ-услуг, образующих ФС ИС. При этом каждая ИТ-услуга может быть описана как продукт, имеющий определенную стоимость. Эта стоимость определяется:

а) стоимостью создания ИТ-услуги, реализующей эту ИТ-услугу совокупности ИТ-сервисов и комплекта документов по каждому из этих ИТ-сервисов;

б) стоимостью адаптации к требованиям Потребителя ИТ-услуги, реализующей эту ИТ-услугу совокупности ИТ-сервисов и комплекта документов по каждому из этих ИТ-сервисов.

Потребитель получает от Поставщика информацию о возможных вариантах конфигурации этих услуг в рамках ИС и стоимости этих вариантов для Потребителя. Каждый из предлагаемых Поставщиком вариантов конфигурации Потребитель оценивает с точки зрения эффекта от внедрения и эксплуатации этого варианта в своих БП. Затем Потребитель выбирает из совокупности вариантов конфигурации ИС тот, который обеспечивает максимальный эффект.

Такое представление позволяет определить основную концепцию управления взаимоотношениями Поставщика и Потребителя как проверенную на практике концепцию управления взаимоотношениями с клиентами (Client Relationship Management, CRM). В то же время необходимо учесть существование сложившегося представления работ по разработке, внедрению, сопровождению и модификации ИС как ИТ-проекта, ориентированного на выполнение требований Потребителей при ограничениях на качество, стоимость создания или адаптации, время создания или адаптации и содержание работ по созданию или адаптации ИС как совокупности ИТ-услуг.

Сказанное выше позволяет описать глобальную цель деятельности Поставщика в процессе проектирования архитектуры следующим образом [89]:

«Целью Поставщика ИТ-услуг является предоставление Потребителю ИТ-услуг такого набора ИТ-услуг, который наилучшим образом соответствует комплексу требований, определенных Потребителем и особенностями БП Потребителя, при ограничениях на стоимость, время выполнения, качество и содержание работ по предоставлению данного набора ИТ-услуг Потребителю».

Глобальную цель деятельности Потребителя в процессе проектирования архитектуры с учетом сказанного выше можно описать следующим образом [89]:

«Целью Потребителя ИТ-услуг является поиск и организация взаимодействия с таким Поставщиком ИТ-услуг, который предоставляет набор ИТ-услуг, наилучшим образом соответствующий комплексу требований, определенных Потребителем и особенностями БП Потребителя, при ограничениях на стоимость, время выполнения, качество и содержание работ по предоставлению данного набора ИТ-услуг, выбранных Поставщиком».

Данное представление глобальных целей деятельности Поставщика и Потребителя ИТ-услуг определяют основные особенности формализованного представления процессов проектирования архитектуры ИС как совокупности ИТ-услуг. Однако прежде необходимо определить способ формального представления процессов, рассмотренных в подразделе 2.2. К началу 2000-х гг. сложилось определенное представление процессных моделей как формализованных описаний системы БП в целом. Данное представление основано на системно-операционном определении понятия «процесс» и, в частности, понятия «бизнес-процесс». Согласно этому определению, БП – это операция, включенная в систему операций, целью которой является производство и поставка услуг/товаров операциям, входящим в систему, а также другим системам [104]. Следствием такого определения является возможность описания процессов разработки ИС и, в частности, процесса проектирования архитектуры целевым функционалом и множеством ограничений [105-107]. Подобное описание, в свою очередь, позволит поставить задачу оптимизации управления ресурсами процесса (в нашем случае – управления требованиями к ИС) и определить способы решения этой задачи.

При этом следует помнить: то, что определяется термином «бизнес-процесс» (или «процесс»), является не действием, выполняемым реально в текущий момент времени, а представлением или, в отдельных случаях, моделью таких действий. Следовательно, эффективность процесса – это идеальная оценка деятельности, получаемая в процессе того или иного моделирования деятельности предприятия [104]. При этом исполнитель процесса (в нашем случае Поставщик или Потребитель ИТ-услуг) является механизмом, поставляющим услуги, обеспечивающие исполнение процесса, то есть исполнитель процесса сам является процессом. Аналогично ресурсы, поставляемые процессу, не являются его составной частью, а являются своего рода ограничениями для исполнения БП.

В системно-операционном определении процесса базовыми категориями являются понятия «действие», «услуга/товар» и «ресурс». Первое из них – это одно из простейших отношений между оперирующей стороной и предметом, которое является интуитивно ясным и, как правило, однозначно интерпретируемым. Под ресурсом обычно понимают реальный объект, существование которого предполагается, а стоимость или ценность которого не возрастает в процессе совершения операций над ним. Снижение ценности ресурса обусловлено, например, износом оборудования, усталостью или старением персонала, а информационный ресурс может не потерять своей ценности в процессе использования. Напротив, стоимость услуги (товара) в процессе ее производства и поставки возрастает. Таким образом, ресурс является источником возникновения услуги (товара). В этом состоит основное отличие ресурса от услуги (товара) на интервале времени исполнения операции. Следует признать, однако, что предлагаемые исходные понятия нуждаются в дополнительной конкретизации [104].

Основным ресурсом процесса проектирования архитектуры ИС на разных уровнях представления ИС являются требования к ИС, выдвинутые Потребителем, зафиксированные и принятые к исполнению Поставщиком. Поэтому в качестве основного показателя, определяющего уровень достижимости глобальных целей Поставщика и Потребителя в процессе проектирования архитектуры ИС, является показатель, характеризующий степень удовлетворения требований к ИС, выдвинутых Потребителем и принятых к исполнению Поставщиком. При этом будем исходить из того, что выдвигаемые Потребителем разнородные требования будут являться элементами множества требований к конкретной ИС Tr_{IS}. В общем случае это множество будет иметь следующий вид:

$$Tr_{IS} = (tr_1, tr_2, ..., tr_i, ..., tr_n), \quad (2.1)$$

где tr_i – обобщенное описание i-го требования к ИС, отдельной ИТ-услуге ИС или же к отдельному ИТ-сервису ИТ-услуги ИС;

i – идентификатор обобщенного описания требования к ИС tr_i, $i = \overline{1,n}$;

n – количество требований, выдвинутых к ИС, ее ИТ-услугам и ИТ-сервисам этих услуг.

Тогда степень удовлетворения каждого требования к ИС tr_i можно в общем случае описать оператором $r(tr_i)$, который ставит в соответствие описанию требования tr_i число в диапазоне [0...1]. При этом ситуация $r(tr_i) = 0$ означает, что требование tr_i было выдвинуто Потребителем, но не было выполнено Поставщиком по тем или иным причинам. Ситуация $r(tr_i) = 1$ означает, что требование tr_i было выдвинуто Потребителем и выполнено Поставщиком полностью. Ситуация $0 < r(tr_i) < 1$ означает, что требование tr_i было выдвинуто Потребителем и выполнено Поставщиком лишь частично.

Для оценки стоимости выполнения требования введем оператор $pay(r(tr_i))$. В случае, если $r(tr_i) = 0$, данный оператор ставит в соответствие описанию требования tr_i нулевое значение. В случае, если $0 < r(tr_i) \le 1$, данный оператор ставит в соответствие описанию требования tr_i положительное значение, характеризующее величину финансовых затрат $pay(r(tr_i))$ на выполнение требования tr_i со степенью удовлетворения $r(tr_i)$.

Для оценки времени выполнения требования введем оператор $t(r(tr_i))$. В случае, если $r(tr_i) = 0$, данный оператор ставит в соответствие описанию требования tr_i нулевое значение. В случае, если $0 < r(tr_i) \le 1$, данный оператор ставит в соответствие описанию требования tr_i положительное значение, характеризующее величину затрат времени $t(r(tr_i))$ на выполнение требования tr_i со степенью удовлетворения $r(tr_i)$.

Для оценки качества выполнения требования введем оператор $q(r(tr_i))$. В случае, если $r(tr_i) = 0$, данный оператор ставит в соответствие описанию требования tr_i нулевое значение. В случае, если $0 < r(tr_i) \le 1$, данный оператор ставит в соответствие описанию требования tr_i положительное значение, характеризующее качество $q(r(tr_i))$ выполнения требования tr_i со степенью удовлетворения $r(tr_i)$.

Для оценки объема работ по выполнению требования введем оператор $w(r(tr_i))$. В случае, если $r(tr_i) = 0$, данный оператор ставит в соответствие описанию требования tr_i нулевое значение. В случае, если $0 < r(tr_i) \le 1$, данный оператор ставит в соответствие описанию требования tr_i положительное значение, формируемое следующим образом:

$$w(r(tr_i)) = \sum_{j=1}^{m} w_j(r(tr_i)) > 0, \quad (2.2)$$

где $w_j(r(tr_i))$ – объем j-й работы по выполнению требования tr_i со степенью удовлетворения $r(tr_i)$;

j – идентификатор работы в общем перечне работ по созданию ИС, отдельной ИТ-услуги или же отдельного ИТ-сервиса;

m – общее количество работ по созданию ИС, отдельной ИТ-услуги или же отдельного ИТ-сервиса.

Введенные обобщенные описания требований и операторы позволяют сформулировать обобщенное формализованное описание глобальной цели Поставщика ИТ-услуг в процессе проектирования архитектуры ИС (в соответствии с положениями стандарта ISO/IEC 15288:2002) как задачу следующего вида:

$$F_{Pr} = \sum_{i=1}^{n} r^{Pr}(tr_i) \to max, \quad (2.3)$$

$$\begin{cases} \sum_{i=1}^{n} \alpha_i^{Pr} pay(r^{Pr}(tr_i)) \geq pay^*(\sum_{i=1}^{n} r^{Pr}(tr_i)); \\ \sum_{i=1}^{n} \beta_i^{Pr} t(r^{Pr}(tr_i)) \leq t^*(\sum_{i=1}^{n} r^{Pr}(tr_i)); \\ \sum_{i=1}^{n} \gamma_i^{Pr} q(r^{Pr}(tr_i)) \geq q^*(\sum_{i=1}^{n} r^{Pr}(tr_i)); \\ \sum_{i=1}^{n} \delta_i^{Pr} w(r^{Pr}(tr_i)) \geq w^*(\sum_{i=1}^{n} r^{Pr}(tr_i)). \end{cases} \quad (2.4)$$

где Pr – обозначение Поставщика ИТ-услуг и соответствующих ИТ-сервисов;

$r^{Pr}(tr_i)$ – степень удовлетворения требования к ИС tr_i с точки зрения Поставщика;

$\alpha_i^{\Pr}$ – нормативный коэффициент стоимости выполнения требования tr_i, учитывающий индивидуальные особенности Поставщика;

$pay^*(\sum_{i=1}^{n} r^{\Pr}(tr_i))$ – минимально допустимая для Поставщика величина стоимости выполнения множества требований к ИС;

$\beta_i^{\Pr}$ – нормативный коэффициент длительности выполнения требования tr_i, учитывающий индивидуальные особенности Поставщика;

$t^*(\sum_{i=1}^{n} r^{\Pr}(tr_i))$ – максимально допустимое для Поставщика время выполнения множества требований к ИС;

$\gamma_i^{\Pr}$ – нормативный коэффициент качества выполнения требования tr_i, учитывающий индивидуальные особенности Поставщика;

$q^*(\sum_{i=1}^{n} r^{\Pr}(tr_i))$ – минимально допустимое для Поставщика качество выполнения множества требований к ИС;

$\delta_i^{\Pr}$ – нормативный коэффициент объема работ по выполнению требования tr_i, учитывающий индивидуальные особенности Поставщика;

$w^*(\sum_{i=1}^{n} r^{\Pr}(tr_i))$ – минимально допустимый для Поставщика перечень работ по выполнению множества требований к ИС.

Обобщенное формализованное описание глобальной цели Потребителя в процессе проектирования архитектуры ИС (в соответствии с положениями стандарта ISO/IEC 15288:2002) может быть представлено задачей следующего вида:

$$F_U = \sum_{i=1}^{n} r^U(tr_i) \to \max, \tag{2.5}$$

$$\begin{cases} \sum_{i=1}^{n} \alpha_i^U \, pay(r^U(tr_i)) \le pay^*(\sum_{i=1}^{n} r^U(tr_i)); \\ \sum_{i=1}^{n} \beta_i^U t(r^U(tr_i)) \le t^*(\sum_{i=1}^{n} r^U(tr_i)); \\ \sum_{i=1}^{n} \gamma_i^U q(r^U(tr_i)) \ge q^*(\sum_{i=1}^{n} r^U(tr_i)); \\ \sum_{i=1}^{n} \delta_i^U w(r^U(tr_i)) \ge w^*(\sum_{i=1}^{n} r^U(tr_i)), \end{cases} \tag{2.6}$$

где U – обозначение Потребителя ИТ-услуг;

$r^U(tr_i)$ – степень удовлетворения требования к ИС tr_i с точки зрения Потребителя;

α_i^U – нормативный коэффициент стоимости выполнения требования tr_i, учитывающий индивидуальные особенности Потребителя;

$pay^*(\sum_{i=1}^{n} r^U(tr_i))$ – максимально допустимая для Потребителя величина стоимости выполнения множества требований к ИС;

β_i^U – нормативный коэффициент длительности выполнения требования tr_i, учитывающий индивидуальные особенности Потребителя;

$t^*(\sum_{i=1}^{n} r^U(tr_i))$ – максимально допустимое для Потребителя время выполнения множества требований к ИС;

γ_i^U – нормативный коэффициент качества выполнения требования tr_i, учитывающий индивидуальные особенности Потребителя;

$q^*(\sum_{i=1}^{n} r^U(tr_i))$ – минимально допустимое для Потребителя качество выполнения множества требований к ИС;

δ_i^U – нормативный коэффициент объема работ по выполнению требования tr_i, учитывающий индивидуальные особенности Потребителя;

$w^*(\sum_{i=1}^{n} r^U(tr_i))$ – минимально допустимый для Потребителя перечень работ по выполнению множества требований к ИС.

Однако такое формализованное представление целей Поставщика и Потребителя имеет излишне общий характер. Поэтому необходимо детально проанализировать природу элементов этого представления для последующей его детализации и практического применения.

Как показано выше, целевые функции (2.3) и (2.5) описывают степень удовлетворения Поставщика и Потребителя соответственно результатами выполнения требований к ИС. Однако детализация представления этих функций усложняется разнородностью требований, выдвигаемых к ИС. В общем случае множество требований Tr_{IS} можно разделить на следующие подмножества:

- подмножество бизнес-требований;
- подмножество требований к ИС как аспекту бизнеса;
- подмножество требований к ИС в целом;
- подмножество функциональных требований к ИТ-услугам;
- подмножество нефункциональных требований к ИТ-услугам;
- подмножество функциональных требований к ИТ-сервисам;

- подмножество нефункциональных требований к ИТ-сервисам.

Исходя из такого представления, множество (2.1) можно представить следующим образом:

$$Tr_{IS} = Tr_{IS}^{B} \cup Tr_{IS}^{IB} \cup Tr_{IS}^{s} \cup Tr_{IS}^{f} \cup Tr_{IS}^{nf} \cup Tr_{IS}^{fw} \cup Tr_{IS}^{nfw}, \quad (2.7)$$

где Tr_{IS}^{B} – подмножество бизнес-требований, в общем случае имеющее вид

$$Tr_{IS}^{B} = (tr_1, tr_2, ..., tr_i, ..., tr_a), a < n; \quad (2.8)$$

Tr_{IS}^{IB} – подмножество требований к ИС как аспекту бизнеса, в общем случае имеющее вид

$$Tr_{IS}^{IB} = (tr_{a+1}, tr_{a+2}, ..., tr_i, ..., tr_b), b < n; \quad (2.9)$$

Tr_{IS}^{s} – подмножество требований к ИС в целом, в общем случае имеющее вид

$$Tr_{IS}^{s} = (tr_{b+1}, tr_{b+2}, ..., tr_i, ..., tr_c), c < n; \quad (2.10)$$

Tr_{IS}^{f} – подмножество функциональных требований к ИТ-услугам, в общем случае имеющее вид

$$Tr_{IS}^{f} = (tr_{c+1}, tr_{c+2}, ..., tr_i, ..., tr_e), e < n; \quad (2.11)$$

Tr_{IS}^{nf} – подмножество нефункциональных требований к ИТ-услугам, в общем случае имеющее вид

$$Tr_{IS}^{nf} = (tr_{e+1}, tr_{e+2}, ..., tr_i, ..., tr_g), g < n; \quad (2.12)$$

Tr_{IS}^{fw} – подмножество функциональных требований к ИТ-сервисам, в общем случае имеющее вид

$$Tr_{IS}^{fw} = (tr_{g+1}, tr_{g+2}, ..., tr_i, ..., tr_k), k < n; \quad (2.13)$$

Tr_{IS}^{nfw} – подмножество нефункциональных требований к ИТ-сервисам, в общем случае имеющее вид

$$Tr_{IS}^{nfw} = (tr_{k+1}, tr_{k+2}, ..., tr_i, ..., tr_n). \quad (2.14)$$

Количество бизнес-требований равно a. Количество требований к ИС как к аспекту БП равно $(b-a)$. Количество требований к ИС в целом равно $(c-b)$. Количество функциональных требований к ИТ-услугам равно $(e-c)$. Количество нефункциональных требований к ИТ-услугам равно $(g-e)$. Количество функциональных требований к ИТ-сервисам равно $(k-g)$. Количество нефункциональных требований к ИТ-сервисам равно $(n-k)$.

Для любых двух подмножеств Tr_{IS}^{x} и Tr_{IS}^{y}, выделенных в множестве требований к ИС (2.7), должно также выполняться следующее условие:

$$Tr_{IS}^{x} \bigcap Tr_{IS}^{y} = \varnothing. \quad (2.15)$$

Это условие подразумевает невозможность рассмотрения описания любого из требований к ИС как принадлежащего одновременно к двум или более группам требований. Иными словами, ни одно требование не может быть, например, описано как функциональное и нефункциональное одновременно.

Тогда целевые функции (2.3) и (2.5) примут, соответственно, следующий вид:

$$F_{Pr} = \sum_{i=1}^{a} r^{Pr}(tr_i^B) + \sum_{i=a+1}^{b} r^{Pr}(tr_i^{IB}) + \sum_{i=b+1}^{c} r^{Pr}(tr_i^s) + \sum_{i=c+1}^{e} r^{Pr}(tr_i^f) + \\ + \sum_{i=e+1}^{g} r^{Pr}(tr_i^{nf}) + \sum_{i=g+1}^{k} r^{Pr}(tr_i^{fw}) + \sum_{i=k+1}^{n} r^{Pr}(tr_i^{nfw}) \to max \quad , \quad (2.16)$$

$$F_U = \sum_{i=1}^{a} r^{U}(tr_i^B) + \sum_{i=a+1}^{b} r^{U}(tr_i^{IB}) + \sum_{i=b+1}^{c} r^{U}(tr_i^s) + \sum_{i=c+1}^{e} r^{U}(tr_i^f) + \\ + \sum_{i=e+1}^{g} r^{U}(tr_i^{nf}) + \sum_{i=g+1}^{k} r^{U}(tr_i^{fw}) + \sum_{i=k+1}^{n} r^{U}(tr_i^{nfw}) \to max \quad . \quad (2.17)$$

Однако подобное представление целевых функций на практике усложняется следующими факторами:

а) ни один представитель Потребителя, как правило, не имеет представления обо всей последовательности операций всех ИТ-услуг ИС;

б) Потребитель принимает решение о целесообразности внедрения и эксплуатации ИС, прежде всего исходя из прагматической ценности ИТ-услуг предлагаемого Поставщиком варианта ИС (в данном случае прагматическая ценность определяется как степень соответствия ИТ-услуг ИС задачам управления БП Потребителя);

в) непосредственные исполнители работ по созданию ИС со стороны Поставщика не обязаны знать особенности БП конкретного Потребителя, непосредственно не влияющие на разработку соответствующих ИТ-услуг;

г) Поставщик принимает решение о целесообразности разработки варианта ИС, исходя из своих возможностей формирования набора ИТ-услуг, выполняющих множество требований Потребителя, при условии выполнения системы ограничений (2.4) в целом с возможностью нарушения отдельных неравенств.

Эти факторы позволяют рассматривать целевые функции (2.16) и (2.17), определяющие решения Поставщика и Потребителя при выполнении процесса проектирования архитектуры ИС, как функции выбора оптимального набора ИТ-услуг ИС при соблюдении Поставщиком и Потребителем определенных ограничений. При этом выполнение Поставщиком остальных групп требований будут относиться к числу дополнительных ограничений, частично установленных Потребителем, а частично устанавливаемых самим Поставщиком в процессе разработки ИС.

Тогда описание глобальной цели Поставщика ИТ-услуг в процессе проектирования архитектуры ИС (в соответствии с положениями стандарта ISO/IEC 15288:2002) можно формализованно представить следующим образом:

$$F_{Pr} = \sum_{i=c+1}^{e} r^{Pr}(tr_i^f) \to max; \qquad (2.18)$$

$$
\begin{cases}
\sum_{i=c+1}^{e} \alpha_i^{Pr} pay(r^{Pr}(tr_i^f)) \geq pay^*(\sum_{i=c+1}^{e} r^{Pr}(tr_i^f)); \\
\sum_{i=c+1}^{e} \beta_i^{Pr} t(r^{Pr}(tr_i^f)) \leq t^*(\sum_{i=c+1}^{e} r^{Pr}(tr_i^f)); \\
\sum_{i=c+1}^{e} \gamma_i^{Pr} q(r^{Pr}(tr_i^f)) \geq q^*(\sum_{i=c+1}^{e} r^{Pr}(tr_i^f)); \\
\sum_{i=c+1}^{e} \delta_i^{Pr} w(r^{Pr}(tr_i^f)) \geq w^*(\sum_{i=c+1}^{e} r^{Pr}(tr_i^f)); \\
\sum_{i=1}^{a} \mu_i^{Pr} r^{Pr}(tr_i^B) \geq \rho_{Pr}^B; \\
\sum_{i=a+1}^{b} \nu_i^{Pr} r^{Pr}(tr_i^{IB}) \geq \rho_{Pr}^{IB}; \\
\sum_{i=b+1}^{c} \chi_i^{Pr} r^{Pr}(tr_i^s) \geq \rho_{Pr}^s; \\
\sum_{i=e+1}^{g} \eta_i^{Pr} r^{Pr}(tr_i^{nf}) \geq \rho_{Pr}^{nf}; \\
\sum_{i=g+1}^{k} \varsigma_i^{Pr} r^{Pr}(tr_i^{fw}) \geq \rho_{Pr}^{fw}; \\
\sum_{i=k+1}^{n} \omega_i^{Pr} r^{Pr}(tr_i^{nfw}) \geq \rho_{Pr}^{nfw},
\end{cases}
\tag{2.19}
$$

где μ_i^{Pr} – нормативный коэффициент полноты реализации бизнес-требования tr_i^B, учитывающий индивидуальные особенности Поставщика;

ρ_{Pr}^B – минимально допустимый для Поставщика уровень полноты реализации подмножества бизнес-требований;

ν_i^{Pr} – нормативный коэффициент полноты реализации требования к ИС как к аспекту бизнеса tr_i^{IB}, учитывающий индивидуальные особенности Поставщика;

ρ_{Pr}^{IB} – минимально допустимый для Поставщика уровень полноты реализации подмножества требований к ИС как к аспекту БП;

χ_i^{Pr} – нормативный коэффициент полноты реализации требования к ИС в целом tr_i^s, учитывающий индивидуальные особенности Поставщика;

ρ_{Pr}^s – минимально допустимый для Поставщика уровень полноты реализации подмножества требований к ИС в целом;

η_i^{Pr} – нормативный коэффициент полноты реализации нефункционального требования к ИТ-услуге tr_i^{nf}, учитывающий индивидуальные особенности Поставщика;

ρ_{Pr}^{nf} – минимально допустимый для Поставщика уровень полноты реализации подмножества нефункциональных требований к ИТ-услугам;

ς_i^{Pr} – нормативный коэффициент полноты реализации функционального требования к ИТ-сервису tr_i^{fw}, учитывающий индивидуальные особенности Поставщика;

ρ_{Pr}^{fw} – минимально допустимый для Поставщика уровень полноты реализации подмножества функциональных требований к ИТ-сервисам;

ω_i^{Pr} – нормативный коэффициент полноты реализации нефункционального требования к ИТ-сервису tr_i^{nfw}, учитывающий индивидуальные особенности Поставщика;

ρ_{Pr}^{nfw} – минимально допустимый для Поставщика уровень полноты реализации подмножества нефункциональных требований к ИТ-сервисам.

Описание глобальной цели Потребителя ИТ-услуг в процессе проектирования архитектуры ИС (в соответствии с положениями стандарта ISO/IEC 15288:2002) можно формализованно представить следующим образом:

$$F_U = \sum_{i=c+1}^{e} r^U(tr_i^f) \to \max; \qquad (2.20)$$

$$\begin{cases} \sum_{i=c+1}^{e} \alpha_i^U \, pay(r^U(tr_i^f)) \geq pay^*(\sum_{i=c+1}^{e} r^U(tr_i^f)); \\ \sum_{i=c+1}^{e} \beta_i^U \, t(r^U(tr_i^f)) \leq t^*(\sum_{i=c+1}^{e} r^U(tr_i^f)); \\ \sum_{i=c+1}^{e} \gamma_i^U \, q(r^U(tr_i^f)) \geq q^*(\sum_{i=c+1}^{e} r^U(tr_i^f)); \\ \sum_{i=c+1}^{e} \delta_i^U \, w(r^U(tr_i^f)) \geq w^*(\sum_{i=c+1}^{e} r^U(tr_i^f)); \\ \sum_{i=1}^{a} \mu_i^U \, r^U(tr_i^B) \geq \rho_U^B; \\ \sum_{i=a+1}^{b} \nu_i^{Pr} \, r^{Pr}(tr_i^{IB}) \geq \rho_{Pr}^{IB}; \\ \sum_{i=b+1}^{c} \chi_i^U \, r^U(tr_i^s) \geq \rho_U^s; \\ \sum_{i=e+1}^{g} \eta_i^U \, r^U(tr_i^{nf}) \geq \rho_U^{nf}; \\ \sum_{i=g+1}^{k} \varsigma_i^U \, r^U(tr_i^{fw}) \geq \rho_U^{fw}; \\ \sum_{i=k+1}^{n} \omega_i^U \, r^U(tr_i^{nfw}) \geq \rho_U^{nfw}, \end{cases} \quad (2.21)$$

где μ_i^U – нормативный коэффициент полноты реализации бизнес-требования tr_i^B, учитывающий индивидуальные особенности Потребителя;

ρ_U^B – минимально допустимый для Потребителя уровень полноты реализации подмножества бизнес-требований;

ν_i^U – нормативный коэффициент полноты реализации требования к ИС как к аспекту бизнеса tr_i^{IB}, учитывающий индивидуальные особенности Потребителя;

ρ_U^{IB} – минимально допустимый для Потребителя уровень полноты реализации подмножества требований к ИС как к аспекту БП;

χ_i^U – нормативный коэффициент полноты реализации требования к ИС в целом tr_i^s, учитывающий индивидуальные особенности Потребителя;

ρ_U^s – минимально допустимый для Потребителя уровень полноты реализации подмножества требований к ИС в целом;

η_i^U – нормативный коэффициент полноты реализации нефункционального требования к ИТ-услуге tr_i^{nf}, учитывающий индивидуальные особенности Потребителя;

ρ_U^{nf} – минимально допустимый для Потребителя уровень полноты реализации подмножества нефункциональных требований к ИТ-услугам;

ς_i^U – нормативный коэффициент полноты реализации функционального требования к ИТ-сервису tr_i^{fw}, учитывающий индивидуальные особенности Потребителя;

ρ_U^{fw} – минимально допустимый для Потребителя уровень полноты реализации подмножества функциональных требований к ИТ-сервисам;

ω_i^U – нормативный коэффициент полноты реализации нефункционального требования к ИТ-сервису tr_i^{nfw}, учитывающий индивидуальные особенности Потребителя;

ρ_U^{nfw} – минимально допустимый для Потребителя уровень полноты реализации подмножества нефункциональных требований к ИТ-сервисам.

Описание глобальной цели Поставщика в процессе проектирования архитектуры ИТ-сервисов ИС (в соответствии с положениями стандарта ISO/IEC 12207:2008) можно формализованно представить следующим образом:

$$F_{Pr} = \sum_{i=g+1}^{k} r^{Pr}(tr_i^{fw}) \to max; \qquad (2.22)$$

$$
\begin{cases}
\sum_{i=c+1}^{e} \alpha_i^{Pr} pay(r^{Pr}(tr_i^f)) \geq pay^*(\sum_{i=c+1}^{e} r^{Pr}(tr_i^f)); \\
\sum_{i=c+1}^{e} \beta_i^{Pr} t(r^{Pr}(tr_i^f)) \leq t^*(\sum_{i=c+1}^{e} r^{Pr}(tr_i^f)); \\
\sum_{i=c+1}^{e} \gamma_i^{Pr} q(r^{Pr}(tr_i^f)) \geq q^*(\sum_{i=c+1}^{e} r^{Pr}(tr_i^f)); \\
\sum_{i=c+1}^{e} \delta_i^{Pr} w(r^{Pr}(tr_i^f)) \geq w^*(\sum_{i=c+1}^{e} r^{Pr}(tr_i^f)); \\
\sum_{i=1}^{a} \mu_i^{Pr} r^{Pr}(tr_i^B) \geq \rho_{Pr}^B; \\
\sum_{i=a+1}^{b} \nu_i^{Pr} r^{Pr}(tr_i^{IB}) \geq \rho_{Pr}^{IB}; \\
\sum_{i=b+1}^{c} \chi_i^{Pr} r^{Pr}(tr_i^s) \geq \rho_{Pr}^s; \\
\sum_{i=c+1}^{e} \theta_i^{Pr} r^{Pr}(tr_i^f) \geq \rho_{Pr}^f; \\
\sum_{i=e+1}^{g} \eta_i^{Pr} r^{Pr}(tr_i^{nf}) \geq \rho_{Pr}^{nf}; \\
\sum_{i=k+1}^{n} \omega_i^{Pr} r^{Pr}(tr_i^{nfw}) \geq \rho_{Pr}^{nfw},
\end{cases}
\tag{2.23}
$$

где θ_i^{Pr} – нормативный коэффициент полноты реализации функционального требования к ИТ-услуге tr_i^f, учитывающий индивидуальные особенности Поставщика;

ρ_{Pr}^f – минимально допустимый для Поставщика уровень полноты реализации подмножества функциональных требований к ИТ-услугам.

Описание глобальной цели Потребителя в процессе проектирования архитектуры ИТ-сервисов ИС (в соответствии с положениями стандарта ISO/IEC 12207:2008) можно формализованно представить следующим образом:

$$F_U = \sum_{i=g+1}^{k} r^U(tr_i^{fw}) \to \max; \tag{2.24}$$

$$\begin{cases} \sum_{i=c+1}^{e} \alpha_i^U \, pay(r^U(tr_i^f)) \geq pay^*(\sum_{i=c+1}^{e} r^U(tr_i^f)); \\ \sum_{i=c+1}^{e} \beta_i^U \, t(r^U(tr_i^f)) \leq t^*(\sum_{i=c+1}^{e} r^U(tr_i^f)); \\ \sum_{i=c+1}^{e} \gamma_i^U \, q(r^U(tr_i^f)) \geq q^*(\sum_{i=c+1}^{e} r^U(tr_i^f)); \\ \sum_{i=c+1}^{e} \delta_i^U \, w(r^U(tr_i^f)) \geq w^*(\sum_{i=c+1}^{e} r^U(tr_i^f)); \\ \sum_{i=1}^{a} \mu_i^U \, r^U(tr_i^B) \geq \rho_U^B; \\ \sum_{i=a+1}^{b} \nu_i^{Pr} \, r^{Pr}(tr_i^{IB}) \geq \rho_{Pr}^{IB}; \\ \sum_{i=b+1}^{c} \chi_i^U \, r^U(tr_i^s) \geq \rho_U^s; \\ \sum_{i=c+1}^{e} \theta_i^U \, r^U(tr_i^f) \geq \rho_U^f; \\ \sum_{i=e+1}^{g} \eta_i^U \, r^U(tr_i^{nf}) \geq \rho_U^{nf}; \\ \sum_{i=k+1}^{n} \omega_i^U \, r^U(tr_i^{nfw}) \geq \rho_U^{nfw}, \end{cases} \quad (2.25)$$

где θ_i^U – нормативный коэффициент полноты реализации функционального требования к ИТ-услуге tr_i^f, учитывающий индивидуальные особенности Потребителя;

ρ_U^f – минимально допустимый для Потребителя уровень полноты реализации подмножества функциональных требований к ИТ-услугам.

Для достижения рассмотренных выше глобальных цели Поставщик должен выполнять прежде всего такие действия:

- разработка новых ИТ-услуг и ИТ-сервисов, удовлетворяющих требования, выдвигаемые Потребителями;
- адаптация разработанных ранее ИТ-услуг и ИТ-сервисов под особенности требований конкретных Потребителей.

Потребитель для достижения поставленных перед ним глобальных целей должен выполнять прежде всего такие действия:

- формулировка требований к ИС, ИТ-услугам и ИТ-сервисам, учитывающим общие и индивидуальные особенности бизнес-процессов объекта автоматизации;
- анализ и выбор ИТ-услуг и ИТ-сервисов, предлагаемых Поставщиками и соответствующих сформулированным требованиям.

При этом между Поставщиком и Потребителем в процессе проектирования архитектуры ИС в общем случае практически отсутствуют какие-либо формы предварительной договоренности, ограничивающие выбор Потребителя или же разработки Поставщика. Это позволяет представить процесс проектирования архитектуры ИС как оптимального набора ИТ-услуг и реализующих эти услуги оптимального набора ИТ-сервисов в виде бескоалиционной игры. В общем случае такая игра для Поставщика и Потребителя будет иметь вид:

$$\Gamma_{IS} = < \{Pr,U\}, \{X_j\}_{j \in \{Pr,U\}}, \{f_j\}_{j \in \{Pr,U\}} >, \qquad (2.26)$$

где Γ_{IS} – обозначение игры Поставщика и Потребителя;

$\{Pr,U\}$ – множество игроков, участвующих в игре Γ_{IS};

$\{X_j\}_{j \in \{Pr,U\}}$ – множество стратегий игры Γ_{IS};

$\{f_j\}_{j \in \{Pr,U\}}$ – множество функций выигрыша игры Γ_{IS}.

Говоря о стратегиях Поставщика и Потребителя, необходимо, прежде всего, отметить особенность, вызванную модульным подходом к разработке ИС. Чистая стратегия Поставщика представляет собой создание такого типового ФМ как стандартного набора ИТ-услуг и соответствующих ИТ-сервисов, который будет соответствовать представлениям различных Потребителей об автоматизации конкретного БП. Чистая стратегия Потребителя представляет собой решение о покупке типового ФМ как стандартного набора ИТ-услуг и соответствующих ИТ-сервисов для автоматизации соответствующего БП. При этом сам БП может подвергаться реинжинирингу или улучшению с целью обеспечения максимального соответствия особенностям типового ФМ ИС.

Однако использование чистых стратегий зачастую приводит к рассогласованию между особенностями автоматизируемого БП и конкретными ИТ-услугами и ИТ-сервисами, образующими типовый ФМ ИС. В результате такого рассогласования возникают отдельные ИТ-услуги и ИТ-сервисы, которые, находясь в составе типового ФМ, не эксплуатируются в рамках соответствующего БП Потребителя.

В связи с этим особый интерес приобретает возможность взаимодействия Поставщика и Потребителя на основе смешанных стратегий. Для Потребителя смешанная стратегия заключается в выборе такого набора ИТ-услуг из

различных типовых ФМ, чтобы степень удовлетворения этими ИТ-услугами требований, выдвинутых к ИС, стремилась к 1. Для Поставщика смешанная стратегия заключается в выборе такого набора ИТ-услуг, который обеспечивает максимальную степень удовлетворения этих требований имеющимися и вновь разрабатываемыми ИТ-сервисами при соблюдении указанных выше условий.

Использование смешанных стратегий должно обеспечить Поставщику и Потребителю в ходе игры выбор такого варианта конфигурации ИТ-услуг, который:

а) соответствует выдвинутым Потребителем требованиям в максимальной степени;

б) приносит Поставщику желаемый доход от повторного использования ИТ-услуг и соответствующих ИТ-сервисов;

в) обеспечивает реализацию непрерывных технологических цепочек сбора, хранения и обработки данных в рамках единой целостной ИС;

г) гарантирует отсутствие ИТ-услуг и ИТ-сервисов, которые не являются обязательными в технологических цепочках сбора, хранения и обработки данных и не соответствует выдвинутым требованиям.

Таким образом, использование теоретико-игрового представления взаимодействия Поставщика и Потребителя позволяет формализовать процедуры формирования и корректировки набора ИТ-услуг и соответствующих ИТ-сервисов, который обеспечивает приемлемую для каждой стороны степень достижения глобальных целей.

2.5 Выводы

Отказ от сложившейся в 1980-х гг. терминологической базы и постоянное изменение смысла вновь вводимых терминов создает одну из основных проблем компьютерных наук. Большинство новых терминов используются в той степени, в которой они становятся понятными специалистам-теоретикам и инженерам-практикам. При этом, естественно, возникают ситуации переосмысления и искажения смысла каждого из терминов. Подобные ситуации усложняются еще и тем, что оригинальные термины являются, как правило, иноязычными и не вполне адекватно воспринимаются носителями других языков.

Решение подобной проблемы заключается, прежде всего, в разработке и постоянном совершенствовании подходов к представлению систем как объектов исследований. Эти подходы, в свою очередь, формируют систему терминов и определений. В данном исследовании основой подобной системы, сформированной под влиянием сервисного подхода к созданию ИС, являются определения понятий «ИТ-услуга» и «ИТ-сервис», которые рассмотрены в начале подраздела 2.1. На основе этих определений стало возможным

уточнение определения архитектуры ИС на различных уровнях представления, а также определения сторон, участвующих в проектах создания ИС. Предложены формулировки понятий «Архитектура ИТ-услуг» и «Архитектура ИТ-сервисов», скорректированы определения терминов «Поставщик ИТ-услуг» и «Потребитель ИТ-услуг», уточнены возможные роли Поставщиков и Потребителей на разных стадиях жизненного цикла ИС.

Сложившаяся в настоящее время ориентация компьютерных наук и программной инженерии на стандартизацию наиболее успешных практик имеет как свои плюсы, так и серьезные минусы. Несомненным плюсом является то, что собираемый и обобщаемый таким образом успешный опыт выполнения различных работ на разных стадиях жизненного цикла программных систем гораздо легче доводится до конкретных разработчиков подобных систем, программистов-прикладников, аналитиков и менеджеров программных проектов. Однако серьезными минусом с научной точки зрения является то, что и создатели, и потребители подобных стандартов проявляют тенденцию замыкаться только на практической стороне проблемы. Иными словами, успешные практики, становясь самоцелью, заслоняют не менее важный вопрос: возможно ли применение данных практик в других частично отличающихся условиях или для построения другой, частично отличающейся программной системы?

Поиск ответа на этот вопрос может идти двумя основными путями. Первый путь также является ориентированным на практическое решение и заключается, как показано в подразделе 1.6, в поиске своеобразных компромиссов между выделенными в качестве стандартов и методологий практик построения программных систем вообще и ИС в частности. Второй путь более сложен, более затратен и предполагает поиск ответов на следующий вопрос: возможно ли создание объективной научной последовательности преобразований представлений ИС и ее компонентов в процессе жизненного цикла ИС максимальной степени общности? Однако даже частные ответы на указанный вопрос, получаемые в результате следования этим путем, являются краеугольными камнями компьютерных наук как таковых. Отказ от этого другого пути в общем случае может привести к тому, что программная инженерия так и останется ремеслом, отражающим частные случаи более общих законов и закономерностей.

В данном разделе как раз и рассматривается попытка перейти с первого пути на второй. При этом старание сохранить успешные результаты, полученные на пути стандартизации накопленных успешных практик, определяет необходимость подробного изучения процессов, непосредственно работающих с требованиями, а также процессов проектирования архитектуры системы, что и было освещено в подразделе 2.2. Показано, что в настоящее время описания этих процессов в стандартах ISO/IEC 15288:2002 и 12207:2008 являются максимально обобщенными, а для их реализации при разработке ИС как разновидности программной системы требуется создание специальных

архитектурных фреймворков. В качестве такого фреймворка была рассмотрена британская методология разработки ИС SSADM.

В результате анализа было показано, что рассмотренные в подразделе 2.2 процессы, непосредственно работающие с требованиями, и процессы проектирования архитектуры ИС (или ее программной реализации) являются одним из наиболее серьезных «узких мест» проекта создания ИС. В результате рассмотрения последовательностей процессов, непосредственно работающих с требованиями и завершающих эти последовательности процессов проектирования архитектуры, было показано, что решение задачи создания архитектуры ИС до сих пор носит интуитивный характер и является одним из основных факторов, замедляющих (а в некоторых случаях – и срывающих) проект создания ИС. Следует также отметить, что методология SSADM как пример архитектурного фреймворка создания ИС практически не рассматривает подобную задачу, что отрицательно сказывается на ее применении при разработке конкретных ИС.

Для решения этой задачи с учетом особенностей сервисного подхода к созданию ИС в подразделе 2.3 было уточнено понятие «требование к ИС», предложены определения основных групп требований к ИС. Показано, что предлагаемые определения требований к ИС и отдельных групп этих требований соответствуют сложившимся представлениям требований к ПО как к более общему классу систем.

Представление ИС как совокупности ИТ-услуг позволяет свести многообразие концепций управления взаимоотношениями Поставщика и Потребителя к проверенной на практике концепции CRM с учетом специфики жизненного цикла ИС. На основе данной концепции, в подразделе 2.4 были сформулированы определения глобальных целей Поставщика и Потребителя. Далее в подразделе были предложены обобщенные формализованные описания глобальных целей Поставщика и Потребителя в процессе проектирования архитектуры ИТ-услуг ИС и процессе проектирования архитектуры ИТ-сервисов ИС, а также определено обобщенное представление взаимодействия Поставщика и Потребителя в процессе проектирования архитектуры ИС как бескоалиционной игры двух игроков.

Несмотря на то, что предлагаемые формализованные описания носят обобщенный характер, они позволяют определить подходы к поиску конкретных решений проблем формирования и анализа требований, а также проектирования архитектуры ИС, которые и будут рассмотрены в следующих разделах.

3 КОНЦЕПЦИЯ ПРЕДСТАВЛЕНИЯ ТРЕБОВАНИЙ К ИНФОРМАЦИОННОЙ СИСТЕМЕ

3.1 Основные положения концепции представления требований к информационной системе

Как показано в подразделе 2.4, требования к ИС являются основным ресурсом, определяющим особенности выполнения процессов проектирования архитектуры. Соответственно, от того, как именно будут описаны требования на разных уровнях представления ИС, зависит эффективность и качество результатов синтеза архитектуры ИС. Поэтому возникает необходимость перед началом описания требований к ИС определить основную концепцию их представления.

Одной из наиболее важных проблем, возникающих при попытках описать требования к ИС формальными способами, является изначальная неполнота требований. Эта неполнота вызвана, прежде всего, стремлением Потребителя получить только те ИТ-услуги и реализующие их ИТ-сервисы, эксплуатация которых обеспечит скорейшее появление прибыли от автоматизируемого процесса. При этом Потребитель не задумывается о необходимости дополнительных ИТ-услуг и соответствующих ИТ-сервисов, необходимых для обеспечения эффективной согласованной работы основных ИТ-услуг[10]. Не стоит сбрасывать со счетов и «забывчивость» Потребителя, возможность слабой ориентации Потребителя в предложениях Поставщика, а также плохое знание Поставщиком особенностей предметной области Потребителя. Кроме того, следует помнить о возможности изменений автоматизируемого процесса с течением времени (например, в результате реинжиниринга или постепенного улучшения).

Решение этой проблемы становится возможным в том случае, если предположить, что Поставщик и Потребитель в каждом конкретном проекте создания ИС имеют дело не с простым множеством требований к ИС, полностью или частично упорядоченным определенным образом, а с более общей совокупностью требований. При этом в такую совокупность должны входить не только требования к ИС, выдвинутые Потребителем и принятые к исполнению Поставщиком, но и так называемые «забытые» (невыявленные или несвоевременно выявленные) требования, требования, которые Поставщик может выдвинуть к ИС, исходя из своего видения предметной области и т.п. Все эти требования могут раньше или позже стать элементами множества

[10] Например, руководство высшего учебного заведения, желая получить быстрый видимый эффект от автоматизации управления учебным процессом, выдвигает в качестве первоочередных требований к разрабатываемой информационной системе требования к созданию ИТ-услуг «Формирование и ведение учебного плана» и «Расписание». При этом требования к таким промежуточным ИТ-услугам, как, например, «Расчет штатов», «Расчет учебной нагрузки», «Распределение учебной нагрузки между преподавателями кафедры» и т.п. руководством первоначально не выдвигаются и не рассматриваются.

требований к ИС (2.7) в ходе очередной итерации процессов, непосредственно работающих с требованиями (см. подраздел 2.2).

Для обозначения и дальнейшего формализованного описания такой расширенной совокупности требований к ИС используем существующее понятие «универсум». Это понятие воспринимается чаще всего как философский термин, обозначающий всю объективную реальность во времени и в пространстве. Согласно Готфриду Лейбницу, универсум – это «множество всех возможных миров», из которых лишь один – наш мир – реален, а все остальные возможно осмыслить только логическим путём, то есть непротиворечивым образом представляя возможные факты или связи вещей. В современном научном мышлении понятие «универсум» всё чаще приобретает смысл фиксированной системы (или систем) объектов, к которым относятся утверждения (высказывания) какой-либо теории. Именно поэтому говорят об «универсуме рассуждения» – логической абстракции, выражающей мир мыслимых объектов, единство многого, поскольку типы изучаемых в какой-либо теории объектов могут быть различны в зависимости от допускаемых средств и задач исследования. Эта логическая абстракция только отчасти обобщает философскую идею универсума. К элементам универсума рассуждения прежде всего относят индивиды, то есть элементарные объекты изучаемой области, а, кроме того, если требуется, все их виды и роды (множества индивидов). Такое широкое понимание универсума вместе с утверждением, что всякое множество является элементом некоторого универсума, приводит к выводу о бесконечности множеств объектов. Другой аспект анализа идей, связанных с универсумом, – различение стандартного и нестандартного, конструктивного и неконструктивного универсума, а также онтологического и гносеологического универсума рассуждения (отнесение последнего непосредственно к теории, а первого – к возможным моделям этой теории) [108].

Однако подобные представления оставляют в стороне самого исследователя, который своими действиями (или бездействием) формирует универсум с использованием конкретных архитектурных фреймворков, методов и методик. В то же время опыт разработки многих программных систем и, в частности, ИС, показывает, что выбор тех или иных инструментальных средств выполнения этих проектов в существенной степени определяется предпочтениями конкретных разработчиков. Аналогичная ситуация возникает, например, при выборе моделей и методов анализа и проектирования программных систем[11] [109].

Исходя из этого, предлагается интерпретировать философско-логическое понятие «универсум» применительно к потребностям разработки ИС следующим образом: «универсум – это «множество всех возможных систем», из которых лишь одна – исследуемая система – реальна, а все остальные (в том

[11] Мартин Фаулер, характеризуя жаркие дебаты между авторами различных методов анализа и проектирования программных систем, приводит старую шутку: «Вопрос: какая разница между автором методологии и террористом? Ответ: с террористом можно договориться».

числе и проектируемую ИС) возможно осмыслить только логическим путем, то есть непротиворечивым образом представляя возможные факты или связи исследуемой системы». Развивая данную интерпретацию, понятие «универсум» в теории и практике создания, внедрения, эксплуатации и модернизации ИС можно определить следующим образом: «***Универсум – совокупность данных, информации и знаний об исследуемой системе, объекте или процессе, как известных, так и неизвестных исследователю, в распоряжении которого имеется конечное множество методов получения и обработки этих данных, информации и знаний».***

Данное определение порождает целый ряд следствий, из которых для формализованного описания требований к ИС и процессов проектирования архитектуры важны, прежде всего:

а) *следствие 1* из данного определения: точность описания исследуемой системы, объекта или процесса стремится к максимуму в том случае, если объем неизвестных исследователю данных, информации и знаний об этой системе, объекте или процессе стремится к минимуму;

б) *следствие 2* из данного определения: исследователь всегда должен допускать, что совокупность неизвестных ему данных, информации и знаний об исследуемой системе, объекте или процессе не является пустой;

в) *следствие 3* из данного определения и следствий 1 и 2: для практического применения потребителем универсум должен обладать не максимальной точностью, а такой, которая позволит на основании совокупности известных данных, информации и знаний принимать решения с желаемыми для потребителя универсума характеристиками эффективности и качества.

Формально предлагаемое определение универсума U может быть описано следующим образом:

$$U = (< D, I, K >, < UnD, UnI, UnK >, (F) >), \qquad (3.1)$$

где U – обозначение универсума;

D – множество данных, известных исследователю благодаря применению имеющихся в его распоряжении методов добычи и обработки данных;

I – множество информации, известной исследователю благодаря применению имеющихся в его распоряжении методов добычи и обработки информации;

K – множество знаний, известных исследователю благодаря применению имеющихся в его распоряжении методов добычи и обработки знаний;

UnD – множество данных, не известных исследователю;

UnI – множество информации, не известной исследователю;

UnK – множество знаний, не известных исследователю;

$(F) :< UnD, UnI, UnK > \rightarrow < D, I, K >$ – множество отображений, описывающих множество методов, преобразующих неизвестные данные, информацию и знания в известные.

Рассмотрим описание совокупности требований к ИС с использованием универсума (3.1). Согласно определению универсума, множество требований к ИС (2.7) становится известным тогда, когда требования, составляющие это множество, выдвинуты Потребителем, приняты к исполнению Поставщиком и согласованы между Поставщиком и Потребителем. В дальнейшем множество (2.7), используемое для описания известных требований к ИС, будем называть множеством сформулированных требований к ИС. Каждое из подмножеств множества сформулированных требований к ИС в этом случае может быть определено следующим образом:

$$Tr_{IS}^{B} =< D_{IS}^{B}, I_{IS}^{B}, K_{IS}^{B} >; Tr_{IS}^{IB} =< D_{IS}^{IB}, I_{IS}^{IB}, K_{IS}^{IB} >; Tr_{IS}^{s} =< D_{IS}^{s}, I_{IS}^{s}, K_{IS}^{s} >;$$
$$Tr_{IS}^{f} =< D_{IS}^{f}, I_{IS}^{f}, K_{IS}^{f} >; Tr_{IS}^{nf} =< D_{IS}^{nf}, I_{IS}^{nf}, K_{IS}^{nf} >; \quad (3.2)$$
$$Tr_{IS}^{fw} =< D_{IS}^{fw}, I_{IS}^{fw}, K_{IS}^{fw} >; Tr_{IS}^{IB} =< D_{IS}^{nfw}, I_{IS}^{nfw}, K_{IS}^{nfw} > .$$

В этих описаниях каждый из элементов кортежей представляет собой описания данных, информации и знаний, определяющих сформулированные требования к ИС.

Исходя из представлений групп сформулированных требований (3.2), представление множества сформулированных требований к ИС Tr_{IS} в универсуме требований будет иметь вид:

$$Tr_{IS} =< D_{IS}, I_{IS}, K_{IS} >=< D_{IS}^{B}, I_{IS}^{B}, K_{IS}^{B} > \cup < D_{IS}^{IB}, I_{IS}^{IB}, K_{IS}^{IB} > \cup$$
$$\cup < D_{IS}^{s}, I_{IS}^{s}, K_{IS}^{s} > \cup < D_{IS}^{f}, I_{IS}^{f}, K_{IS}^{f} > \cup < D_{IS}^{nf}, I_{IS}^{nf}, K_{IS}^{nf} > \cup \quad (3.3)$$
$$\cup < D_{IS}^{fw}, I_{IS}^{fw}, K_{IS}^{fw} > \cup < D_{IS}^{nfw}, I_{IS}^{nfw}, K_{IS}^{nfw} > .$$

Что касается данных, информации и знаний, неизвестных одному или всем участникам проекта создания ИС, то они могут быть описаны следующим образом:

$$< UnD, UnI, UnK >=< UnD^{Sp}, UnI^{Sp}, UnK^{Sp} > \cup$$
$$\cup < UnD^{Cs}, UnI^{Cs}, UnK^{Cs} > \cup < UnD^{Sp\&Cs}, UnI^{Sp\&Cs}, UnK^{Sp\&Cs} >, \quad (3.4)$$

где $< UnD^{Sp}, UnI^{Sp}, UnK^{Sp} >$ – множество данных, информации и знаний, не известных Поставщику;

$< UnD^{Cs}, UnI^{Cs}, UnK^{Cs} >$ – множество данных, информации и знаний, не известных Потребителю;

$< UnD^{Sp\&Cs}, UnI^{Sp\&Cs}, UnK^{Sp\&Cs} >$ – множество данных, информации и знаний, не известных Поставщику и Потребителю.

Множество методов добычи и обработки для универсума требований можно в этом случае разделить на следующие подмножества:

а) подмножество методов выявления и анализа требований Потребителем, имеющее в общем случае следующий вид:

$$(F_{Sp}^{Sp\&Cs}) :< UnD^{Sp\&Cs}, UnI^{Sp\&Cs}, UnK^{Sp\&Cs} > \rightarrow \\ \rightarrow < UnD^{Sp}, UnI^{Sp}, UnK^{Sp} >; \quad (3.5)$$

б) подмножество методов выявления и анализа требований Поставщиком, имеющее в общем случае следующий вид:

$$(F_{Cs}^{Sp\&Cs}) :< UnD^{Sp\&Cs}, UnI^{Sp\&Cs}, UnK^{Sp\&Cs} > \rightarrow \\ \rightarrow < UnD^{Cs}, UnI^{Cs}, UnK^{Cs} >; \quad (3.6)$$

в) подмножество методов согласования требований, выявленных Потребителем и предъявленных Поставщику, в общем случае имеющее вид:

$$(F^{Sp}) :< UnD^{Sp}, UnI^{Sp}, UnK^{Sp} > \rightarrow < D, I, K >; \quad (3.7)$$

г) подмножество методов согласования требований, выявленных Поставщиком и предъявленных Потребителю, в общем случае имеющее вид:

$$(F^{Cs}) :< UnD^{Cs}, UnI^{Cs}, UnK^{Cs} > \rightarrow < D, I, K >. \quad (3.8)$$

Тогда универсум структуры требований к ИС в общем случае будет иметь вид:

$$U_{Tr_{IS}} = (< D_{IS}, I_{IS}, K_{IS} >, < UnD^{Sp}, UnI^{Sp}, UnK^{Sp} >, \\ < UnD^{Cs}, UnI^{Cs}, UnK^{Cs} >, < UnD^{Sp\&Cs}, UnI^{Sp\&Cs}, UnK^{Sp\&Cs} >, \\ (F_{Sp}^{Sp\&Cs}), (F_{Cs}^{Sp\&Cs}), (F^{Sp}), (F^{Cs})). \quad (3.9)$$

Данное описание основано на модели, а точнее, на иерархии DIKW (Data, Information, Knowledge and Wisdom), предложенной для управления знаниями [110]. Использование этой иерархии для описания процессов, непосредственно работающих с требованиями, а также процессов

проектирования архитектуры обусловлено сложившимся в 1990-х гг. убеждением [111], согласно которому никакое представление требований в одном виде не дает их полной картины. Сравнение представлений требований, полученных различными специалистами в ходе разнообразных исследований, помогает выявить несоответствия, неясности, допущения и упущения, которые трудно обнаружить, когда требования представлены в одном формате.

Следует отметить, что классическая модель DIKW предполагает четкую направленность преобразований данных в информацию, информации – в знания, а знаний – в мудрость. Однако в процессах, непосредственно работающих с требованиями, часто основой для моделирования требований к ИС является представление этих требований на уровне информации, выраженное в виде их текстовых описаний или же визуальных моделей (см. рис. 3.1). При этом процесс проектирования архитектуры системы предполагает возможное уточнение визуальных моделей требований описанием этих требований на уровне данных в виде той или иной атрибутивной модели.

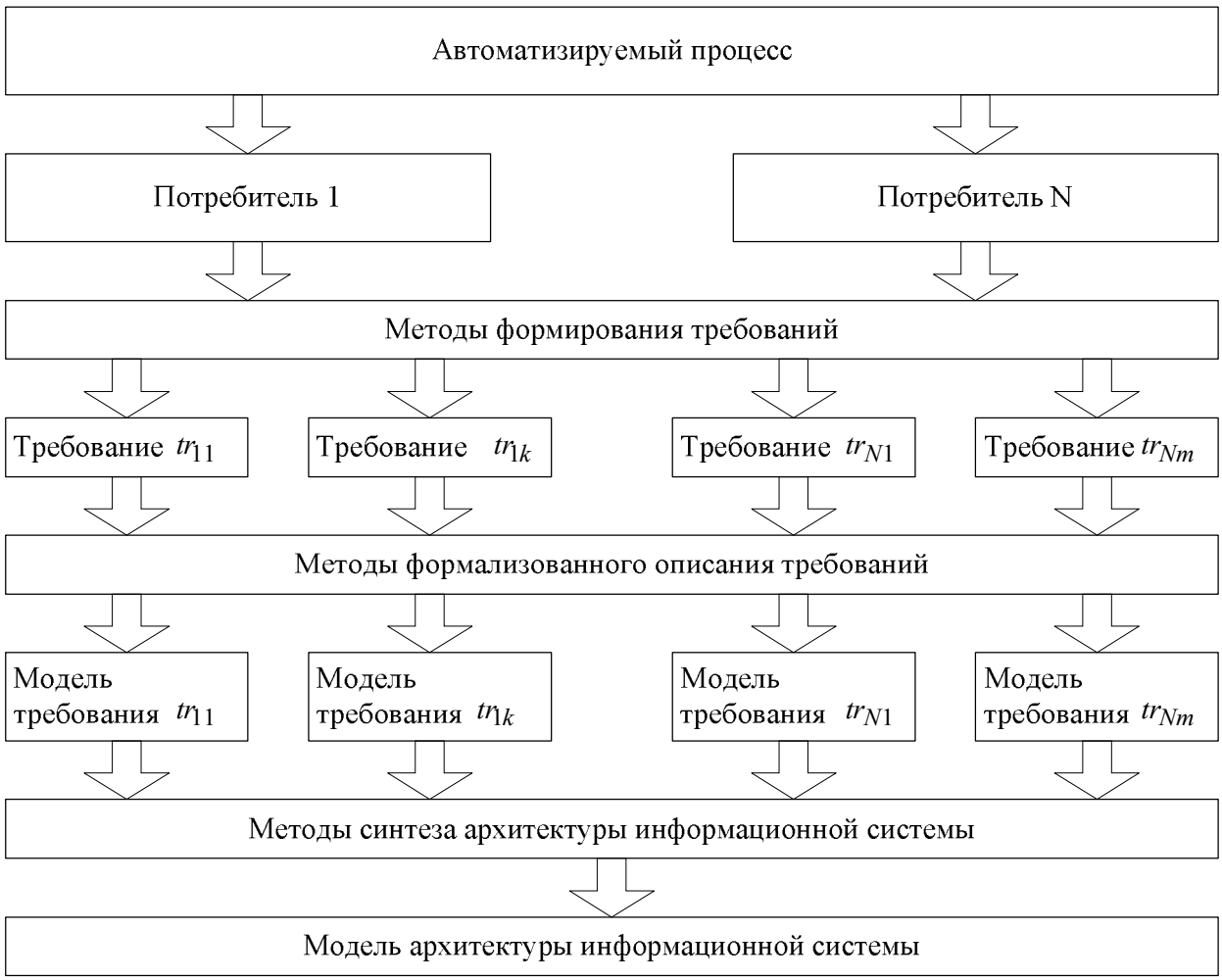

Рисунок 3.1 – Схема последовательности преобразований описаний требований к информационной системе при проектировании архитектуры системы

В то же время использование атрибутивных моделей, рассмотренных в подразделе 1.5 или аналогичных им, для представления требований к ИС на уровне данных затруднительно. Это затруднение вызвано, прежде всего, стремлением объединить в рамках одной атрибутивной модели требования к ИС наборы атрибутов, используемых в принципиально разных целях[12].

Поэтому возникает необходимость рассмотрения каждого требования к ИС как изначального многообразия представлений этого требования в виде данных, информации и знаний. Существование такого многообразия обусловлено следующими причинами:

а) представление требований к ИС на уровне информации предназначено для описания различными способами элементов автоматизируемых объектов или процессов, элементов разрабатываемой ИС или ИС в целом с целью обеспечения возможности выполнения проекта создания ИС, к которой выдвинуто требование путем выполнения последовательности преобразований «неформализованное описание требования на естественном языке – частично формализованные описания требований – формальное описание требования в виде набора целевых показателей, значения которых характеризуют степень удовлетворения требования $r(tr_i)$»;

б) представление требований на уровне данных предназначено для формирования описаний требования к ИС и его отдельных представлений, взаимно согласованных друг с другом, с целью обеспечения возможности осуществления стандартных операций ввода, отображения, редактирования и удаления требования к ИС и его отдельных представлений в рамках ИТ формирования и анализа требований к ИС, а также управления отдельными требованиями в ходе выполнения проекта создания ИС;

в) представление требования к ИС на уровне знаний предназначено для выявления знаний об автоматизируемых объектах или процессах, разрабатываемых элементах ИС или ИС в целом с целью обеспечения возможности повторного использования этих знаний в проектах создания других ИС.

Принятие рассмотрения каждого требования к ИС как изначального многообразия его представлений в качестве аксиомы формирования универсума требований (3.1) приводит к тому, что начальным описанием любого возможного требования к ИС, как известного, так и неизвестного, может быть любой из следующих вариантов:

а) описание требования к ИС в виде данных (например, фрагмент атрибутивной модели требования к ИС, однозначно определяющей

[12] Среди этих целей по результатам исследования атрибутивных моделей требований, приведенным в подразделе 1.5, можно выделить: адресацию требования в каталоге требований или любом другом хранилище представлений требований; установление взаимосвязей между отдельными требованиями, в том числе – между требованиями различных групп; управление требованиями; описания нефункциональных требований как дополнительных характеристик функциональных требований; верификацию и валидацию требования.

идентификаторы повторно используемого представления требования к ИС или всего требования в целом);

б) описание требования к ИС в виде информации (например, результат интервьюирования Потребителя ИТ-услуг или же разработанная Поставщиком ИТ-услуг модель потоков данных);

в) описание требования к ИС в виде знаний (например, онтология предметной области, сформулированная Поставщиком на основе своего опыта автоматизации конкретного типа БП или же Потребителем на основе изучения своих БП).

Подобное представление универсума требований к ИС позволяет пересмотреть процесс проектирования архитектуры системы. Если рассматривать данный процесс с точки зрения методологии SSADM и ей подобных, то последовательность преобразований требований к ИС можно описать схемой, показанной на рис. 3.1 [112]. При этом методология SSADM рассматривает модель архитектуры ИС как структурную модель потоков данных и дополняющие ее структурные модели данных ИС и ее отдельных функций.

Такая последовательность преобразований требований к ИС требует от Потребителей максимально полной и четкой формулировки требований, чтобы на основе этой формулировки можно было построить точную и непротиворечивую модель этих требований. Иными словами, полученную от Потребителей информацию о требованиях к ИС Поставщик в ходе выполнения процессов формирования и анализа требований должен преобразовать в структурные модели, описывающие наборы данных, процессы их ввода, хранения, обработки и отображения, а также движение этих наборов данных между процессами, внешними сущностями и хранилищами данных.

Однако общие особенности подобных моделей определяются особенностями моделируемого объекта. Это значит, что особенности моделей архитектуры системы (например, моделей ФС ИС) должны отражать особенности моделей автоматизируемых БП предприятия. Следовательно, и модели требований как исходные данные для построения модели архитектуры системы, и формулировки требований, выдвинутых Потребителями и принятых к исполнению Поставщиком, должны отражать особенности процессного подхода к описанию объекта управления, системы управления (а в нашем случае – еще и ИС как элемента этой системы управления).

Как уже отмечалось в подразделе 2.4, подход к представлению требований в виде описаний процессов был сформулирован еще в 1993 г. [102]. С учетом предложенного в стандарте ISO/IEC 24774:2007 подхода к описанию процесса минимальная процессная атрибутивная модель требования к ИС должна включать в себя следующие атрибуты:

а) название (наименование) процесса, к которому выдвигается требование;

б) ожидаемые результаты выполнения процесса, к которому выдвигается требование (выходы требования);

в) виды деятельности, выполняемые в рамках процесса, к которому выдвигается требование.

Основываясь на опыте моделирования процессов [41–43, 104], в дополнение к указанным атрибутам минимальная процессная атрибутивная модель требования к ИС может также включать следующие атрибуты:

а) цели, достижение которых свидетельствует о выполнении требования;

б) ресурсы, обрабатываемые процессом, к которому выдвигается требование (входы требования);

в) неизменяемые ресурсы, используемые процессом, к которому выдвигается требование (механизмы выполнения требования);

г) перечень Потребителей и Поставщиков, которые выдвигают и принимают к исполнению требование (источники требования).

Как и рассмотренные в подразделе 2.2 описания процессов, непосредственно работающих с требованиями, данная минимальная процессная атрибутивная модель требования является своеобразным паттерном, определяющим основные особенности более детальных атрибутивных моделей требований к ИС. Для использования этой атрибутивной модели при разработке ИС Поставщик и Потребитель должны детализировать ее применением конкретной методологии работы с требованиями к ИС и управления ими.

Сказанное выше определяет концепцию представления требований к ИС как набор следующих положений [113]:

а) отказ от рассмотрения только множества сформулированных требований к ИС и изначальное представление требований к ИС как элементов универсума, включающего в себя как известные, так и неизвестные Поставщику, Потребителю или им обоим требования к ИС, а также методы формирования этих требований;

б) изначальное многообразие представлений требований к ИС в виде данных, информации и знаний;

в) процессный подход к описанию требований, определяющий минимальную процессную атрибутивную модель требования к ИС;

г) подход к управлению требованиями к ИС, основанный на изложенном в подразделе 1.6 основном принципе управления требованиями.

Изложенная концепция определяет основные особенности формализованного описания групп требований к ИС и отдельных требований, их моделей различного вида, методов и средств формирования, анализа и управления этими требованиями, а также возможных моделей и методов синтеза описания проектируемой архитектуры ИС на различных уровнях представления.

3.2 Формализованное описание универсума требований к информационной системе

Сформулированные в подразделе 3.1 основные положения концепции представления требований к ИС определяют общие формализованные описания требований. Учитывая положения а) и б) данной концепции, общие формализованные описания требований к ИС должны включать в себя:

а) формализованные описания как известных, так и не известных Поставщику, Потребителю или им обоим требований к ИС;

б) формализованные описания методов формирования требований к ИС;

в) формализованные описания представлений требований к ИС в виде данных, информации и знаний.

Кроме того, с учетом предложенного в подразделе 3.1 представления универсума требований к ИС в виде выражения (3.9), общие формализованные описания требований к ИС дополнительно должны включать в себя:

а) формализованные описания методов, приемов и способов преобразования представлений различных требований, выполненных на одном и том же уровне (то есть представлений требований к ИС в виде данных, информации или знаний), самих в себя и друг в друга, что позволит установить и отображать взаимосвязи между отдельными требованиями группы, обусловленные особенностями автоматизируемого процесса;

б) формализованные описания методов, приемов и способов преобразования представлений требований к ИС, выполненных на одном уровне, в представления требований, выполненных на другом уровне (например, преобразования представлений требований к ИС, выполненных в виде информации, в представления требований к ИС, выполненных в виде данных).

Рассмотренные условия создания формализованных описаний требований к ИС заставляют отказаться от классических теоретико-множественных описаний требований. Математический аппарат, на основе которого следует разрабатывать формализованные описания групп требований к ИС, должен позволять:

- рассматривать требования к ИС как единую целостную систему – прообраз разрабатываемой ИС – и одновременно как множества связанных друг с другом элементов, преобразуемые одно в другое в соответствии с методологией и технологиями формирования и анализа требований;

- учитывать влияние на представления требований к ИС конкретных методов, приемов и способов формирования и преобразования представлений этих требований.

Данным условиям лучше всего отвечает математический аппарат теории категорий [114-117].

Рассмотрим категорные модели элементов универсума требований. Так модель множества данных, информации и знаний, неизвестных Поставщику и Потребителю $L_{Sp\&Cs}$, будет иметь вид:

$$L_{Sp\&Cs} = [UnD^{Sp\&Cs}, UnI^{Sp\&Cs}, UnK^{Sp\&Cs}, H(UnD^{Sp\&Cs}), H(UnI^{Sp\&Cs}), H(UnK^{Sp\&Cs}), H(UnD^{Sp\&Cs}, UnI^{Sp\&Cs}), H(UnI^{Sp\&Cs}, UnD^{Sp\&Cs}), H(UnD^{Sp\&Cs}, UnK^{Sp\&Cs}), H(UnK^{Sp\&Cs}, UnD^{Sp\&Cs}), H(UnI^{Sp\&Cs}, UnK^{Sp\&Cs}), H(UnK^{Sp\&Cs}, UnI^{Sp\&Cs})], \quad (3.10)$$

где $UnD^{Sp\&Cs}$ – подкласс представлений требований, не известных Поставщику и Потребителю, в виде данных;

$UnI^{Sp\&Cs}$ – подкласс представлений требований, не известных Поставщику и Потребителю, в виде информации;

$UnK^{Sp\&Cs}$ – подкласс представлений требований, не известных Поставщику и Потребителю, в виде знаний;

$H(UnD^{Sp\&Cs})$ – подмножество морфизмов, описывающих преобразования представлений требований, не известных Поставщику и Потребителю, в виде данных самих в себя и друг в друга, имеющее следующий вид:

$$\forall X, Y \in UnD^{Sp\&Cs} \quad H(UnD^{Sp\&Cs}) = H(1_X) \cup H(X, Y) \cup H(1_Y); \quad (3.11)$$

$1_X, 1_Y$ – единичные морфизмы, условия существования которых описаны в [114];

$H(UnI^{Sp\&Cs})$ – подмножество морфизмов, описывающих преобразования представлений требований, не известных Поставщику и Потребителю, в виде информации самих в себя и друг в друга, имеющее следующий вид:

$$\forall X, Y \in UnI^{Sp\&Cs} \quad H(UnI^{Sp\&Cs}) = H(1_X) \cup H(X, Y) \cup H(1_Y); \quad (3.12)$$

$H(UnK^{Sp\&Cs})$ – подмножество морфизмов, описывающих преобразования представлений требований, не известных Поставщику и Потребителю, в виде знаний самих в себя и друг в друга, имеющее следующий вид:

$$\forall X, Y \in UnK^{Sp\&Cs} \quad H(UnK^{Sp\&Cs}) = H(1_X) \cup H(X, Y) \cup H(1_Y); \quad (3.13)$$

$H(UnD^{Sp\&Cs}, UnI^{Sp\&Cs})$ – подмножество морфизмов, описывающих методы, приемы и способы преобразования представлений требований, не известных Поставщику и Потребителю, в виде данных в представления требований, неизвестных Поставщику и Потребителю, в виде информации;

$H(UnI^{Sp\&Cs}, UnD^{Sp\&Cs})$ – подмножество морфизмов, описывающих методы, приемы и способы преобразования представлений требований, не известных Поставщику и Потребителю, в виде информации в представления требований, неизвестных Поставщику и Потребителю, в виде данных;

$H(UnD^{Sp\&Cs}, UnK^{Sp\&Cs})$ – подмножество морфизмов, описывающих методы, приемы и способы преобразования представлений требований, не известных Поставщику и Потребителю, в виде данных в представления требований, неизвестных Поставщику и Потребителю, в виде знаний;

$H(UnK^{Sp\&Cs}, UnD^{Sp\&Cs})$ – подмножество морфизмов, описывающих методы, приемы и способы преобразования представлений требований, не известных Поставщику и Потребителю, в виде знаний в представления требований, неизвестных Поставщику и Потребителю, в виде данных;

$H(UnI^{Sp\&Cs}, UnK^{Sp\&Cs})$ – подмножество морфизмов, описывающих методы, приемы и способы преобразования представлений требований, не известных Поставщику и Потребителю, в виде информации в представления требований, неизвестных Поставщику и Потребителю, в виде знаний;

$H(UnK^{Sp\&Cs}, UnI^{Sp\&Cs})$ – подмножество морфизмов, описывающих методы, приемы и способы преобразования представлений требований, не известных Поставщику и Потребителю, в виде знаний в представления требований, неизвестных Поставщику и Потребителю, в виде информации.

Рассмотренные особенности модели (3.10) определяют особенности категорных моделей, описывающих кортежи $<UnD^{Sp}, UnI^{Sp}, UnK^{Sp}>$ и $<UnD^{Cs}, UnI^{Cs}, UnK^{Cs}>$. Так модель множества данных, информации и знаний, не известных Поставщику, $<UnD^{Sp}, UnI^{Sp}, UnK^{Sp}>$ будет иметь следующий вид:

$$L_{Sp} = [UnD^{Sp}, UnI^{Sp}, UnK^{Sp}, H(UnD^{Sp}), H(UnI^{Sp}), H(UnK^{Sp}),$$
$$H(UnD^{Sp}, UnI^{Sp}), H(UnI^{Sp}, UnD^{Sp}), H(UnD^{Sp}, UnK^{Sp}), \quad (3.14)$$
$$H(UnK^{Sp}, UnD^{Sp}), H(UnI^{Sp}, UnK^{Sp}), H(UnK^{Sp}, UnI^{Sp})].$$

Модель множества данных, информации и знаний, не известных Потребителю, $<UnD^{Cs}, UnI^{Cs}, UnK^{Cs}>$ будет иметь следующий вид:

$$L_{Cs} = [UnD^{Cs}, UnI^{Cs}, UnK^{Cs}, H(UnD^{Cs}), H(UnI^{Cs}), H(UnK^{Cs}),$$
$$H(UnD^{Cs}, UnI^{Cs}), H(UnI^{Cs}, UnD^{Cs}), H(UnD^{Cs}, UnK^{Cs}),$$
$$H(UnK^{Cs}, UnD^{Cs}), H(UnI^{Cs}, UnK^{Cs}), H(UnK^{Cs}, UnI^{Cs})]. \quad (3.15)$$

Для описания множества сформулированных требований к ИС (2.7), представленного в универсуме требований к ИС в виде выражения (3.3), предлагается каждую группу требований рассматривать как отдельную категорию. Так модель бизнес-требований к ИС L_{IS}^{B} будет иметь следующий вид:

$$L_{IS}^{B} = [D_{IS}^{B}, I_{IS}^{B}, K_{IS}^{B}, H(D_{IS}^{B}), H(I_{IS}^{B}), H(K_{IS}^{B}), H(D_{IS}^{B}, I_{IS}^{B}),$$
$$H(I_{IS}^{B}, D_{IS}^{B}), H(D_{IS}^{B}, K_{IS}^{B}), H(K_{IS}^{B}, D_{IS}^{B}), H(I_{IS}^{B}, K_{IS}^{B}), H(K_{IS}^{B}, I_{IS}^{B})], \quad (3.16)$$

где D_{IS}^{B} – подкласс представлений сформулированных бизнес-требований к ИС в виде данных;

I_{IS}^{B} – подкласс представлений сформулированных бизнес-требований к ИС в виде информации;

K_{IS}^{B} – подкласс представлений сформулированных бизнес-требований к ИС в виде знаний;

$H(D_{IS}^{B})$ – подмножество морфизмов, описывающих преобразования представлений сформулированных бизнес-требований к ИС в виде данных самих в себя и друг в друга, имеющее следующий вид:

$$\forall X, Y \in D_{IS}^{B} \quad H(D_{IS}^{B}) = H(1_X) \cup H(X, Y) \cup H(1_Y); \quad (3.17)$$

$1_X, 1_Y$ – единичные морфизмы, условия существования которых описаны в [114];

$H(I_{IS}^{B})$ – подмножество морфизмов, описывающих преобразования представлений сформулированных бизнес-требований к ИС в виде информации самих в себя и друг в друга, имеющее следующий вид:

$$\forall X, Y \in I_{IS}^{B} \quad H(I_{IS}^{B}) = H(1_X) \cup H(X, Y) \cup H(1_Y); \quad (3.18)$$

$H(K_{IS}^{B})$ – подмножество морфизмов, описывающих преобразования представлений сформулированных бизнес-требований к ИС в виде знаний самих в себя и друг в друга, имеющее следующий вид:

$$\forall X, Y \in K_{IS}^{B} \quad H(K_{IS}^{B}) = H(1_X) \cup H(X, Y) \cup H(1_Y); \tag{3.19}$$

$H(D_{IS}^{B}, I_{IS}^{B})$ – подмножество морфизмов, описывающих методы, приемы и способы преобразования представлений сформулированных бизнес-требований к ИС в виде данных в представления сформулированных бизнес-требований к ИС в виде информации;

$H(I_{IS}^{B}, D_{IS}^{B})$ – подмножество морфизмов, описывающих методы, приемы и способы преобразования представлений сформулированных бизнес-требований к ИС в виде информации в представления сформулированных бизнес-требований к ИС в виде данных;

$H(D_{IS}^{B}, K_{IS}^{B})$ – подмножество морфизмов, описывающих методы, приемы и способы преобразования представлений сформулированных бизнес-требований к ИС в виде данных в представления сформулированных бизнес-требований к ИС в виде знаний;

$H(K_{IS}^{B}, D_{IS}^{B})$ – подмножество морфизмов, описывающих методы, приемы и способы преобразования представлений сформулированных бизнес-требований к ИС в виде знаний в представления сформулированных бизнес-требований к ИС в виде данных;

$H(I_{IS}^{B}, K_{IS}^{B})$ – подмножество морфизмов, описывающих методы, приемы и способы преобразования представлений сформулированных бизнес-требований к ИС в виде информации в представления сформулированных бизнес-требований к ИС в виде знаний;

$H(K_{IS}^{B}, I_{IS}^{B})$ – подмножество морфизмов, описывающих методы, приемы и способы преобразования представлений сформулированных бизнес-требований к ИС в виде знаний в представления сформулированных бизнес-требований к ИС в виде информации.

Рассмотренные особенности модели (3.16) определяют особенности моделей, описывающих остальные группы требований к ИС. Так модель группы требований к ИС как к аспекту бизнеса L_{IS}^{IB} будет иметь следующий вид:

$$\begin{aligned} L_{IS}^{IB} = [&D_{IS}^{IB}, I_{IS}^{IB}, K_{IS}^{IB}, H(D_{IS}^{IB}), H(I_{IS}^{IB}), H(K_{IS}^{IB}), H(D_{IS}^{IB}, I_{IS}^{IB}), \\ &H(I_{IS}^{IB}, D_{IS}^{IB}), H(D_{IS}^{IB}, K_{IS}^{IB}), H(K_{IS}^{IB}, D_{IS}^{IB}), H(I_{IS}^{IB}, K_{IS}^{IB}), H(K_{IS}^{IB}, I_{IS}^{IB})]. \end{aligned} \tag{3.20}$$

Модель группы требований к ИС в целом L_{IS}^{s} будет иметь следующий вид:

$$L_{IS}^{s} = [D_{IS}^{s}, I_{IS}^{s}, K_{IS}^{s}, H(D_{IS}^{s}), H(I_{IS}^{s}), H(K_{IS}^{s}), H(D_{IS}^{s}, I_{IS}^{s}), H(I_{IS}^{s}, D_{IS}^{s}), H(D_{IS}^{s}, K_{IS}^{s}), H(K_{IS}^{s}, D_{IS}^{s}), H(I_{IS}^{s}, K_{IS}^{s}), H(K_{IS}^{s}, I_{IS}^{s})]. \quad (3.21)$$

Модель группы функциональных требований к ИТ-услугам L_{IS}^{f} будет иметь следующий вид:

$$L_{IS}^{f} = [D_{IS}^{f}, I_{IS}^{f}, K_{IS}^{f}, H(D_{IS}^{f}), H(I_{IS}^{f}), H(K_{IS}^{f}), H(D_{IS}^{f}, I_{IS}^{f}), H(I_{IS}^{f}, D_{IS}^{f}), H(D_{IS}^{f}, K_{IS}^{f}), H(K_{IS}^{f}, D_{IS}^{f}), H(I_{IS}^{f}, K_{IS}^{f}), H(K_{IS}^{f}, I_{IS}^{f})]. \quad (3.22)$$

Модель группы нефункциональных требований к ИТ-услугам L_{IS}^{nf} будет иметь следующий вид:

$$L_{IS}^{nf} = [D_{IS}^{nf}, I_{IS}^{nf}, K_{IS}^{nf}, H(D_{IS}^{nf}), H(I_{IS}^{nf}), H(K_{IS}^{nf}), H(D_{IS}^{nf}, I_{IS}^{nf}), H(I_{IS}^{nf}, D_{IS}^{nf}), H(D_{IS}^{nf}, K_{IS}^{nf}), H(K_{IS}^{nf}, D_{IS}^{nf}), H(I_{IS}^{nf}, K_{IS}^{nf}), H(K_{IS}^{nf}, I_{IS}^{nf})]. \quad (3.23)$$

Модель группы функциональных требований к ИТ-сервисам L_{IS}^{fw} будет иметь следующий вид:

$$L_{IS}^{fw} = [D_{IS}^{fw}, I_{IS}^{fw}, K_{IS}^{fw}, H(D_{IS}^{fw}), H(I_{IS}^{fw}), H(K_{IS}^{fw}), H(D_{IS}^{fw}, I_{IS}^{fw}), H(I_{IS}^{fw}, D_{IS}^{fw}), H(D_{IS}^{fw}, K_{IS}^{fw}), H(K_{IS}^{fw}, D_{IS}^{fw}), H(I_{IS}^{fw}, K_{IS}^{fw}), H(K_{IS}^{fw}, I_{IS}^{fw})]. \quad (3.24)$$

Модель группы нефункциональных требований к ИТ-сервисам L_{IS}^{nfw} будет иметь следующий вид:

$$L_{IS}^{nfw} = [D_{IS}^{nfw}, I_{IS}^{nfw}, K_{IS}^{nfw}, H(D_{IS}^{nfw}), H(I_{IS}^{nfw}), H(K_{IS}^{nfw}), H(D_{IS}^{nfw}, I_{IS}^{nfw}), H(I_{IS}^{nfw}, D_{IS}^{nfw}), H(D_{IS}^{nfw}, K_{IS}^{nfw}), H(K_{IS}^{nfw}, D_{IS}^{nfw}), H(I_{IS}^{nfw}, K_{IS}^{nfw}), H(K_{IS}^{nfw}, I_{IS}^{nfw})]. \quad (3.25)$$

Основываясь на моделях (3.16). (3.20) – (3.25), можно представить формализованное описание множества сформулированных требований к ИС (3.3) моделью $M_{Tr_{IS}}$, имеющей следующий вид:

$$M_{Tr_{IS}} = [\, L^{B}_{IS}, L^{IB}_{IS}, L^{s}_{IS}, L^{f}_{IS}, L^{nf}_{IS}, L^{fw}_{IS}, L^{nfw}_{IS}, \Phi^{B}_{IB}, \Phi^{IB}_{s}, \Phi^{s}_{f}, \Phi^{s}_{nf}, \Phi^{f}_{nf}, \Phi^{f}_{fw}, \Phi^{nf}_{nfw}, \Phi^{fw}_{nfw}], \tag{3.26}$$

где Φ^{B}_{IB} – одноместный ковариантный функтор [114], устанавливающий связь между категорией L^{B}_{IS} (начало функтора) и категорией L^{IB}_{IS} (конец функтора);

Φ^{IB}_{s} – одноместный ковариантный функтор, устанавливающий связь между категорией L^{IB}_{IS} (начало функтора) и категорией L^{s}_{IS} (конец функтора);

Φ^{s}_{f} – одноместный ковариантный функтор, устанавливающий связь между категорией L^{s}_{IS} (начало функтора) и категорией L^{f}_{IS} (конец функтора);

Φ^{s}_{nf} – одноместный ковариантный функтор, устанавливающий связь между категорией L^{s}_{IS} (начало функтора) и категорией L^{nf}_{IS} (конец функтора);

Φ^{f}_{nf} – одноместный ковариантный функтор, устанавливающий связь между категорией L^{f}_{IS} (начало функтора) и категорией L^{nf}_{IS} (конец функтора);

Φ^{f}_{fw} – одноместный ковариантный функтор, устанавливающий связь между категорией L^{f}_{IS} (начало функтора) и категорией L^{fw}_{IS} (конец функтора);

Φ^{nf}_{nfw} – одноместный ковариантный функтор, устанавливающий связь между категорией L^{nf}_{IS} (начало функтора) и категорией L^{nfw}_{IS} (конец функтора);

Φ^{fw}_{nfw} – одноместный ковариантный функтор, устанавливающий связь между категорией L^{fw}_{IS} (начало функтора) и категорией L^{nfw}_{IS} (конец функтора).

Тогда весь универсум структуры требований к ИС может быть формализованно описан моделью $M_{U_{Tr}}$, имеющей следующий вид [118]:

$$M_{U_{Tr}} = [\, L_{Tr_{IS}}, L_{Sp}, L_{Cs}, L_{Sp\&Cs}, \Phi^{Sp\&Cs}_{Pr}, \Phi^{Sp\&Cs}_{U}, \Phi^{Sp}_{Tr_{IS}}, \Phi^{Cs}_{Tr_{IS}}], \tag{3.27}$$

где $\Phi_{Sp}^{Sp\&Cs}$ – одноместный ковариантный функтор, устанавливающий связь между категорией $L_{Sp\&Cs}$ (начало функтора) и категорией L_{Sp} (конец функтора);

$\Phi_{Cs}^{Sp\&Cs}$ – одноместный ковариантный функтор, устанавливающий связь между категорией $L_{Sp\&Cs}$ (начало функтора) и категорией L_{Cs} (конец функтора);

$\Phi_{Tr_{IS}}^{Sp}$ – одноместный ковариантный функтор, устанавливающий связь между категорией L_{Sp} (начало функтора) и категорией $L_{Tr_{IS}}$ (конец функтора);

$\Phi_{Tr_{IS}}^{Cs}$ – одноместный ковариантный функтор, устанавливающий связь между категорией L_{Cs} (начало функтора) и категорией $L_{Tr_{IS}}$ (конец функтора).

Ковариантность функторов, присутствующих в моделях (3.26) и (3.27), призвана обеспечить в категории-конце функтора сохранение направленности морфизмов, связывающих объекты категории-начала функтора [114]. Данное свойство функторов с прикладной точки зрения обеспечивает сохранность особенностей связей между отдельными требованиями в группе требований к ИС, обусловленными особенностями автоматизируемого процесса.

С прикладной точки зрения модель универсума требований (3.27) является формализованным описанием фрагмента допустимых для реализации архитектурных фреймворков и, в частности, методологий проектирования ИС, в котором рассматриваются возможные модели и методы формирования и анализа требований к ИС. Под словами «допустимые для реализации архитектурные фреймворки и, в частности, методологии проектирования ИС» здесь и в дальнейшем следует понимать то множество фреймворков и, в частности, методологий проектирования ИС, которые могут быть разработаны (или адаптированы) для создания ИС на основе рассмотренной выше концепции представления требований к ИС.

Например, если на основе модели универсума требований (3.27) Поставщик и Потребитель в ходе выполнения проекта создания конкретной ИС утвердят использование архитектурного фреймворка, который может быть описан моделью $M_{M_{Tr}}$ вида

$$M_{M_{Tr}} = [\, L_{Tr_{IS}}, L_{Sp}, L_{Cs}, L_{Sp\&Cs}, \Phi_{Cs}^{Sp\&Cs}, \Phi_{Tr_{IS}}^{Cs} \,], \qquad (3.28)$$

то это будет означать, что выбран архитектурный фреймворк разработки ИС, который предполагает выдвижение требований к ИС Потребителем с последующим принятием Поставщиком выдвинутых Потребителем требований к исполнению [118]. Примером такого архитектурного фреймворка является уже упоминавшаяся выше методология SSADM.

В то же время, если на основе модели универсума требований (3.27) Поставщик и Потребитель в ходе выполнения проекта создания конкретной ИС утвердят использование архитектурного фреймворка, который может быть описан моделью $M_{M_{Tr}}$ вида

$$M_{M_{Tr}} = [\, L_{Tr_{IS}}, L_{Sp}, L_{Cs}, L_{Sp\&Cs}, \Phi_{Sp}^{Sp\&Cs}, \Phi_{Tr_{IS}}^{Sp} \,], \qquad (3.29)$$

то это будет означать, что выбран архитектурный фреймворк разработки ИС, который предполагает выдвижение требований к ИС Поставщиком с последующим утверждением Потребителем выдвинутых Поставщиком требований [118]. Примером такого архитектурного фреймворка является, например, методология быстрого прототипирования функций ИС или же методология, предполагающая использование методов экстремального программирования.

Модель сформулированных требований (3.26) с прикладной точки зрения является формализованным описанием набора допустимых для реализации ИТ формирования и анализа требований к сервис-ориентированным ИС. В соответствии с приведенным в подразделе 1.1 определением понятия «информационная технология», данная модель содержит выполненные в виде функторов описания методов, приемов и способов применения комплекса средств автоматизации или его отдельных элементов, используемых для преобразования представлений сформулированных групп требований к ИС в группы функциональных и нефункциональных требований к ИТ-услугам или в группы функциональных и нефункциональных требований к ИТ-сервисам, реализующим эти ИТ-услуги. Формирование в рамках данной модели групп функциональных и нефункциональных требований к ИТ-услугам предоставит исходную информацию для выполнения процесса проектирования архитектуры ИС. Формирование в рамках данной модели групп функциональных и нефункциональных требований к ИТ-сервисам предоставит исходную информацию для выполнения процесса проектирования архитектуры программного обеспечения ИС (или же программного и информационного обеспечений ИС).

3.3 Обобщенное формализованное описание одноместного ковариантного функтора

Использование функторов в моделях множества сформулированных требований (3.26) и универсума требований (3.27) ставит вопрос о возможности разработки единого подхода к их формализованному описанию. Такой подход важен не только с теоретической, но и с прикладной точки зрения, поскольку его существование в теории ставит перед специалистами проблему

стандартизации и унификации методов, приемов и способов применения комплекса средств автоматизации или его отдельных элементов, используемых для преобразования представлений сформулированных групп требований к ИС в группы функциональных и нефункциональных требований к ИТ-услугам или в группы функциональных и нефункциональных требований к ИТ-сервисам, реализующим эти ИТ-услуги.

Для разработки общего формализованного описания одноместных ковариантных функторов, связывающих две модели в рамках моделей (3.26) и (3.27), обозначим исходную модель как категорию A, а конечную категорную модель как категорию B. Тогда отображение, переводящее исходную категорию A в конечную категорию B, можно будет рассматривать в виде обобщенного одноместного ковариантного функтора Φ_B^A – частного случая отображения категории A в категорию B при выполнении условий [114]:

$$\forall a \in Ob^A \quad \exists \Phi_B^A(a) \in Ob^B \ ; \tag{3.30}$$

$$\forall \alpha \in H_A(a_i, a_j) \subseteq Mor^A \ \exists \Phi_B^A(\alpha) \in H_B(\Phi_B^A(a_i), \Phi_B^A(a_j)) \in Mor^B \ ; \tag{3.31}$$

$$\forall 1_a \in Mor^A \ \exists \Phi_B^A(1_a) = 1_{\Phi_B^A(a)} \in Mor^B \ ; \tag{3.32}$$

$$\begin{gathered} \forall \alpha \in H_A(a_i, a_j) \in Mor^A, \beta \in H_A(a_j, a_k) \in Mor^A \\ \exists \Phi_B^A(\alpha\beta) = \Phi_B^A(\alpha)\Phi_B^A(\beta) \in Mor^B, \end{gathered} \tag{3.33}$$

где Ob^{A} – класс объектов исходной категории A, описывающих элементы класса объектов исходной категорной модели;

a – любой объект, относящийся к классу объектов Ob^A исходной категории A;

Ob^B – класс объектов конечной категории B, описывающих элементы класса объектов конечной категорной модели;

Mor^A – класс морфизмов исходной категории A, описывающих связи элементов класса объектов исходной категорной модели;

$H_A(a_i, a_j)$ – множество морфизмов, определенных в исходной категории A для объекта a_i как начала морфизма и объекта a_j как конца морфизма, при этом $i \neq j$ являются идентификаторами объектов исходной категории A;

α – морфизм, являющийся элементом множества $H_A(a_i, a_j)$;

Mor^B – класс морфизмов конечной категории B, описывающих связи элементов класса объектов конечной категорной модели;

$H_B(\Phi_B^A(a_i),\Phi_B^A(a_j))$ – множество морфизмов, определенных в конечной категории B для объекта $\Phi_B^A(a_i)$ как начала морфизма и объекта $\Phi_B^A(a_j)$ как конца морфизма;

1_a – единичный морфизм, определенный для объекта a исходной категории A;

$1_{\Phi_B^A(a)}$ – единичный морфизм, определенный для соответствующего объекта конечной категории B;

$\alpha\beta$ – частичная бинарная операция умножения морфизмов исходной категории A.

Условия (3.30) – (3.33) позволяют утверждать, что определение обобщенного одноместного ковариантного функтора Φ_B^A целесообразно разделить на две связанные между собой задачи [119]:

- определение системы правил трансформации объектов исходной категории A в объекты конечной категории B;
- определение системы правил трансформации морфизмов исходной категории A в морфизмы конечной категории B.

Эти задачи в работе [120] решены для обобщенного ковариантного функтора, описывающего процесс проектирования распределенных баз данных ИС, причем решение приведено в виде обобщенного алгоритма. Однако этот алгоритм сформулирован с учетом единой природы категорных представлений ИС и ее базы данных. В общем же случае для проектирования ИС одновременно могут использоваться описания требований, в основе которых находятся как структурный, так и объектно-ориентированный подходы.

Описать функтор – это значит:

- описать классы объектов и морфизмов исходной категории A;
- описать классы объектов и морфизмов конечной категории B;
- описать совокупность правил отображения объектов исходной категории A в объекты конечной категории B, удовлетворяющих условию (3.30);
- описать совокупность правил отображения морфизмов исходной категории A в морфизмы конечной категории B, удовлетворяющих условиям (3.31) – (3.33).

Тогда, в общем случае, любой одноместный ковариантный функтор можно описать выражением вида [119, 121]

$$\Phi_B^A = (Ob^A, Ob^B, Mor^A, Mor^B, \Phi_{Ob^A}^{Ob^B}, \Phi_{Mor^A}^{Mor^B}), \qquad (3.34)$$

где $\Phi_{Ob^A}^{Ob^B}$ – система правил трансформации элементов класса объектов исходной категории *A* в элементы класса объектов конечной категории *B*;

$\Phi_{Mor^A}^{Mor^B}$ – система правил трансформации элементов класса морфизмов исходной категории *A* в элементы класса морфизмов конечной категории *B*.

Подобное формализованное описание одноместного ковариантного функтора определяет следующее условие его реализуемости [119]:

$$Ob^A \subseteq \Phi_{Ob^B}^{Ob^A}(Ob^A) \subseteq Ob^B, \quad Mor^A \subseteq \Phi_{Mor^B}^{Mor^A}(Mor^A) \subseteq Mor^B. \quad (3.35)$$

Данное условие для моделей требований к ИС можно интерпретировать следующим образом: фрагмент тезауруса, образующий исходную модель, не должен превышать по размерам и сложности фрагмент тезауруса, образующий конечную модель. Иными словами, при формулировании требований к ИС все многообразие синтаксических конструкций и возможных синонимов, используемых Потребителем для описания некоего понятия предметной области, должно сводиться Поставщиком к единому понятному для него представлению этого понятия в виде данных, информации или знаний. При этом размеры указанных фрагментов тезауруса определяются количеством типов объектов соответствующих категорий. Сложность фрагментов тезауруса определяется количеством типов морфизмов соответствующих категорий.

Подобная интерпретация будет справедливой только для тех случаев, когда уровни представления требований к ИС исходной и конечной категорий будут адекватны друг другу. На практике чаще всего встречается иная ситуация. Так от моделей, описывающих БП предприятия, разработчики переходят к моделям, описывающим ИС управления этим предприятием, а от них, в свою очередь, – к моделям, описывающим отдельные виды обеспечений ИС [41, 42]. Возможны и обратные переходы – например, в ходе реверс-инжиниринга (обратного проектирования) базы данных ИС [122]. Поэтому существование систем правил $\Phi_{Ob^A}^{Ob^B}$ и $\Phi_{Mor^A}^{Mor^B}$ возможно только при выполнении одного из переходов, показанных коммутативными диаграммами

$$\begin{array}{ccc} Tez(Ob^A) & \rightarrow & Tez(Ob^B) \\ \downarrow & & \downarrow \\ MTez(Ob^A) & \rightarrow & MTez(Ob^B) \end{array}; \quad \begin{array}{ccc} Tez(Mor^A) & \rightarrow & Tez(Mor^B) \\ \downarrow & & \downarrow \\ MTez(Mor^A) & \rightarrow & MTez(Mor^B) \end{array}; \quad (3.36)$$

$$\begin{array}{ccc} mTez(Ob^A) & \rightarrow & mTez(Ob^B) \\ \downarrow & & \downarrow \\ Tez(Ob^A) & \rightarrow & Tez(Ob^B) \end{array} ; \quad \begin{array}{ccc} mTez(Mor^A) & \rightarrow & mTez(Mor^B) \\ \downarrow & & \downarrow \\ Tez(Mor^A) & \rightarrow & Tez(Mor^B) \end{array}, \quad (3.37)$$

где $Tez(\bullet)$ – текущий уровень представления фрагмента тезауруса, используемого моделью;

$MTez(\bullet)$ – макроуровень представления фрагмента тезауруса, используемого моделью, для которого $Tez(\bullet)$ является частным случаем;

$mTez(\bullet)$ – микроуровень представления фрагмента тезауруса, используемого моделью, который является частным случаем $Tez(\bullet)$.

Тогда одноместный ковариантный функтор (3.34) можно представить следующим образом [119, 121]:

$$\Phi_B^A = (Sc_{Tez(A)}, Ob^A, Mor^A, Sc_{Tez(B)}, Ob^B, Mor^B, \Phi_{Ob^A}^{Ob^B}, \Phi_{Mor^A}^{Mor^B}), \quad (3.38)$$

где $Sc_{Tez(A)}$ − уровень представления фрагмента тезауруса, используемого исходной моделью требований к ИС, которая описана категорией A;

$Sc_{Tez(B)}$ − уровень представления тезауруса, используемого итоговой моделью требований к ИС, которая описана категорией B.

Выражение (3.38) позволяет сделать вывод о существовании двух основных способов реализации предлагаемого формализованного описания одноместного ковариантного функтора. Первый способ предполагает поиск таких систем правил, которые однозначно определяли бы соответствие заранее заданных элементов исходной и конечной категорий. Второй способ предполагает формирование таких объектов и морфизмов категории B, которые удовлетворяли бы заранее заданным системам правил трансформации, объектам и морфизмам исходной категории A, а также учитывали бы разность уровней представления тезаурусов моделей $Sc_{Tez(A)}$ и $Sc_{Tez(B)}$.

С прикладной точки зрения, первый способ реализации функтора предполагает формирование и реализацию некоего общего алгоритма или базы правил преобразования исходного представления требования к ИС в конечное представление этого же требования или множества других требований. При этом предполагается, что текстовые конструкции или элементы визуальных моделей, описывающие эти требования, заранее известны алгоритму или базе правил и практически не меняются с течением времени. Поэтому данный способ наиболее подходит для таких описаний, которые сводят к минимуму количество элементов алфавита языка описания или моделирования требований к ИС. К подобным описаниям можно отнести большинство распространенных

визуальных моделей ИС и ее компонентов (диаграммы потоков данных, диаграммы «сущность – связь», диаграммы классов и т.п.).

Второй способ реализации функторов предполагает создание и постоянное развитие некоего множества алгоритмов или некоей системы правил преобразования исходного представления требования к ИС в конечное представление этого же требования или множества других требований. При этом темпы изменения этого множества алгоритмов или систем правил определяются темпами появления новых или изменения существующих понятий тезауруса исходного представления требования к ИС. Появление каждого нового, ранее неизвестного понятия в таком представлении требования к ИС (в виде данных, информации или знаний) требует дополнения существующего варианта реализации функтора новыми алгоритмами или правилами, устанавливающими способ и форму описания такого нового понятия в конечном представлении этого требования.

3.4 Использование паттернов проектирования при работе с требованиями к информационной системе

Как следует из моделей сформулированных требований к ИС (3.26) и универсума требований к ИС (3.27), процессы разработки ИС, непосредственно работающие с требованиями, могут рассматриваться как последовательные преобразования представлений требований в виде данных, информации и знаний в ходе выявления и формулирования требований к ИС в соответствии с выбранной методологией разработки ИС. Такие преобразования в предложенных моделях показаны в виде различных вариантов одноместного ковариантного функтора, модель которого (3.38) приведена в подразделе 3.3. Однако эта модель оставляет открытым вопрос о виде представлений отдельных требований к ИС, методах и способах формирования этих представлений и преобразований этих представлений самих в себя и друг в друга.

В общем случае формализованные описания представлений требований к ИС определяются сформулированными в подразделе 3.1 основными положениями концепции представления требований к ИС. Однако кроме указанных положений, на подход к формализованным описаниям представлений требований к ИС в значительной степени влияет стремление Поставщика повторно использовать требования, сформулированные в ходе разработки конкретной ИС, в последующих проектах аналогичных ИС. Подобное стремление приводит к необходимости рассматривать требования к конкретной ИС как частные случаи неких общих шаблонов, описывающих автоматизируемый процесс и разрабатываемую ИС на более высоких уровнях абстракции.

Наиболее близкой к такому взгляду на требования к ИС является идея паттернов проектирования как шаблонов, определяющих решение отдельных

задач, часто повторяющихся в различных проектах программных систем. Эта идея получила свое практическое воплощение в объектно-ориентированном программировании в 1990-х гг. В дальнейшем идея паттернов продолжала развиваться и прочно заняла свою нишу в программной инженерии.

Можно утверждать, что значительным шагом вперед в развитии идей применения шаблонов для проектирования программных систем был сделан в работе [123], первое издание которой вышло в свет в 1995 г. Ее авторы – Э. Гамма, Р. Хелм, Р. Джонсон и Д. Влиссидес – подходят к рассмотрению паттернов проектирования как к результату документирования опыта разработчиков объектно-ориентированных программ и считают, что паттерны упрощают повторное использование удачных проектных и архитектурных решений. Паттерн проектирования именует, абстрагирует и идентифицирует ключевые аспекты структуры общего решения, которые и позволяют применить его для создания повторно используемого дизайна. Он вычленяет участвующие классы и экземпляры, их роль и отношения, а также функции.

При описании каждого паттерна в [123] акцентируется внимание на конкретных задачах объектно-ориентированного проектирования. Подобные описания основаны на представлениях взаимодействий объектов и классов, адаптированных для решения общей задачи проектирования в конкретном контексте, и предполагают использование следующей атрибутивной модели [123]:

а) название и классификация паттерна;

б) назначение;

в) другие распространенные имена;

г) мотивация;

д) применимость;

е) структура;

ж) участники;

и) отношения;

к) результаты;

л) реализация паттерна (точнее, особенности реализации, советы и рекомендуемые приемы реализации);

м) пример кода;

н) известные применения;

о) родственные паттерны.

Для упорядочения описаний паттернов, построенных на основе приведенной атрибутивной модели, в [123] вводится классификация, разделяющая паттерны по их назначению (порождающие, структурные, поведения) и уровню применения (класс, объект). Порождающие паттерны связаны с созданием экземпляров объектов; все они обеспечивают средства логической изоляции клиента от создаваемых объектов. Структурные паттерны объединяют классы или объекты в более крупные структуры. Поведенческие паттерны относятся к взаимодействиям и распределению обязанностей между классами и объектами. Паттерны классов описывают определение отношений

между классами посредством наследования. Отношения между паттернами классов определяются на уровне компиляции. Паттерны объектов описывают отношения между объектами, прежде всего, относящиеся к композиции. Отношения в паттернах объектов обычно определяются на стадии выполнения, а следовательно, обладают большей динамичностью и гибкостью [124].

Однако в [123] отмечается, что данная классификация не является единственно верной и возможно применение других систем классификации паттернов в зависимости от особенностей решаемых задач. В то же время признается, что многие из существующих паттернов могут быть отнесены к нескольким категориям [124].

Следует заметить, что своего рода аналогом паттерна может считаться практически любой алгоритм. Различие заключается в том, что алгоритмы являются шаблонами выполнения вычислительных операций, а не структур данных и методов обработки их значений. Однако и алгоритмы, и классические паттерны проектирования объектно-ориентированного программного обеспечения предполагают некое абстрактное описание проблемы и способа ее решения, которое может быть реализовано в терминах различных языков программирования [124].

Подводя итоги рассмотрению конкретных паттернов, Э. Гамма, Р. Хелм, Р. Джонсон и Д. Влиссидес делают следующий вывод: «Паттерны дают проектировщикам единую терминологию, которой можно пользоваться для общения, документирования и изучения возможных альтернатив. Система начинает выглядеть менее сложной, поскольку о ней становится возможным говорить на более высоком уровне абстракции, чем нотация языка проектирования или программирования». Авторы также предполагают возможность использования паттернов для создания специализированных CASE-средств.

Однако с течением времени стали выявляться основные ограничения использования паттернов в процессе создания программных систем. Прежде всего, эти ограничения связаны со сложностями, которые могут возникнуть при реализации паттернов в конкретном проекте программной системы. Можно утверждать, что критерий простоты использования, который основан на сравнении сложности разрабатываемого программного кода, является основным критерием применимости паттернов проектирования. Однако принятие решения о целесообразности использования паттернов на основе этого критерия является в настоящее время неформализованной процедурой. Кроме того, на принятие такого решения могут повлиять результаты анализа возможных последствий применения паттернов.

Результаты анализа опыта применения паттернов проектирования, рассмотренные в [124], позволяют сделать следующие выводы:

а) паттерны выгодны в случае, если создаваемая программная система должна быть гибкой и предполагает постоянные изменения своей структуры и содержания (добавление или изменение функций и т.п.);

б) паттерны выгодны в случае проведения рефакторинга или других процедур, направленных на совершенствование организации программной системы или ее отдельных элементов [125-128];

в) паттерны невыгодны в случаях, когда затраты ресурсов на создание и сопровождение программной системы, обусловленные сложностью этой системы, превышают эффект от гибкости этой системы.

Основываясь на этих выводах, можно описать современную точку зрения на паттерны как на естественное обобщение результатов процесса проектирования программных систем. При этом применение паттернов проектирования должно являться не самоцелью, а необходимостью, вызванной особенностями конкретного проекта (или портфеля проектов) [124]. Иными словами, использование паттернов проектирования не должно определять архитектурные решения конкретных проектов. Данная точка зрения позволяет, в свою очередь, предположить, что идея использования паттернов в процессах проектирования программных систем является стремлением формализовать и использовать с прикладными целями знания разработчиков таких систем о методах, способах и стилях программирования тех или иных проблем практического характера. Такое предположение хорошо иллюстрируется примером паттернов GRASP. Этой аббревиатурой (англ. General Responsibility Assignment Software Patterns – общие образцы распределения обязанностей) обозначают паттерны, используемые в объектно-ориентированном проектировании для решения общих задач по назначению обязанностей классам и объектам. Приведенные в [129] девять примеров GRASP – Information Expert (Информационный эксперт), Creator (Создатель), Controller (Контроллер), Low Coupling (Слабая связанность), High Cohesion (Сильное зацепление), Polymorphism (Полиморфизм), Pure Fabrication (Чистая выдумка), Indirection (Посредник) и Protected Variations (Сокрытие реализации) – призваны помочь решить отдельные проблемы объектно-ориентированного анализа и проектирования программных систем. Можно сказать, что GRASP-паттерны – это хорошо документированные, стандартизированные и проверенные временем принципы объектно-ориентированного анализа и проектирования, а не попытка привнести что-то принципиально новое [129].

Однако рассмотренная концепция паттернов проектирования ориентирована только на разработку программного обеспечения систем. Использование этой концепции в других процессах жизненного цикла систем (согласно ISO/IEC 15288:2002) без серьезных изменений затруднительно. Особенно это касается процессов разработки ИС как разновидности программных систем. Так, в соответствии с приведенным в подразделе 1.1 описанием главной цели деятельности ИС, одной из основных проблем разработки ИС является проблема отображения желаемого для Потребителя ИТ-услуг единого целостного информационного представления объекта или процесса как на уровне ИТ-услуг (ФС ИС), так и на уровне отдельных ИТ-сервисов (обеспечивающая часть ИС). Для решения данной проблемы

предлагается из всего множества паттернов проектирования в соответствии с предложенной категорной моделью универсума требований к ИС (3.27) выделить такую разновидность паттернов, как паттерны проектирования требований к ИС, используя следующее определение.

Паттерн проектирования требований к ИС – это результат выделения и повторного использования с прикладными целями Поставщиком ИТ-услуг следующих видов знаний [130]:

а) об условии или возможности, которые необходимы Потребителю ИТ-услуг для решения стоящей перед ним проблемы или достижения поставленной перед ним цели;

б) об условии или возможности, которой должна обладать ИС или компонент ИС (ИТ-услуга, ИТ-сервис) с точки зрения Поставщика или Потребителя ИТ-услуг, соответствующих договору, стандарту, спецификации или другому официальному документу;

в) о документированном (с использованием естественного или формального языка) представлении условия или возможности, подобно описанным в первых двух определениях.

В соответствии с предлагаемым определением, паттерны проектирования требований к ИС можно классифицировать следующим образом:

а) по назначению – структурные и поведенческие паттерны;

б) по уровню применения – паттерны представления требований к ИС в виде знаний, паттерны представления требований к ИС в виде информации, паттерны представления требований к ИС в виде данных.

Определения каждого вида паттернов проектирования требований к ИС, выполненные с учетом приведенного выше определения и концепции представления требований к ИС, приведены в табл. 3.1.

Предложенное представление паттернов проектирования требований к ИС позволяет утверждать, что рассмотренные выше модели (3.16), (3.20) – (3.25) являются частными случаями математической модели паттерна проектирования сформулированных требований к ИС M_{IS}^{Pt}, которая имеет следующий вид:

$$M_{IS}^{Pt} = [D_{IS}^{Pt}, I_{IS}^{Pt}, K_{IS}^{Pt}, H(D_{IS}^{Pt}), H(I_{IS}^{Pt}), H(K_{IS}^{Pt}), H(D_{IS}^{Pt}, I_{IS}^{Pt}), H(I_{IS}^{Pt}, D_{IS}^{Pt}), H(D_{IS}^{Pt}, K_{IS}^{Pt}), H(K_{IS}^{Pt}, D_{IS}^{Pt}), H(I_{IS}^{Pt}, K_{IS}^{Pt}), H(K_{IS}^{Pt}, I_{IS}^{Pt})], \quad (3.39)$$

где D_{IS}^{Pt} – подкласс структурных и поведенческих паттернов представлений требований к ИС в виде данных;

I_{IS}^{Pt} – подкласс структурных и поведенческих паттернов представлений требований к ИС в виде информации;

K_{IS}^{Pt} – подкласс структурных и поведенческих паттернов представлений требований к ИС в виде знаний;

Таблица 3.1 – Определения видов паттернов проектирования требований к информационной системе в соответствии с предложенной классификацией

Вид паттерна	Структурный паттерн	Поведенческий паттерн
1	2	3
Паттерн представления требований к ИС в виде знаний	Паттерн знаний о требуемых структурах данных, формирующих единое целостное информационное представление объекта или процесса	Паттерн знаний о процесссе обработки структур данных, формирующих единое целостное информационное представление объекта или процесса
Паттерн представления требований к ИС в виде информации	Паттерн выполненного на естественном или формальном языке описания требования к структурам данных, формирующих единое целостное информационное представление объекта или процесса	Паттерн выполненного на естественном или формальном языке описания требования к процессу обработки структур данных, формирующих единое целостное информационное представление объекта или процесса
Паттерн представления требований к ИС в виде данных	Паттерн ИТ-сервисов и их элементов, обеспечивающих реализацию в рамках требуемой ИТ-услуги структур данных, формирующих единое целостное представление объекта или процесса	Паттерн ИТ-сервисов и их элементов, обеспечивающих реализацию в рамках требуемой ИТ-услуги процесса обработки структур данных, формирующих единое целостное представление объекта или процесса

$H(D_{IS}^{Pt})$ – подмножество морфизмов, описывающих преобразования структурных и поведенческих паттернов представлений требований к ИС в виде данных самих в себя и друг в друга, имеющее следующий вид:

$$\forall X,Y \in D_{IS}^{Pt} \quad H(D_{IS}^{Pt}) = H(1_X) \bigcup H(X,Y) \bigcup H(1_Y); \qquad (3.40)$$

$1_X, 1_Y$ – единичные морфизмы, условия существования которых описаны в [114];

$H(I_{IS}^{Pt})$ – подмножество морфизмов, описывающих преобразования структурных и поведенческих паттернов представлений требований к ИС в виде информации самих в себя и друг в друга, имеющее следующий вид:

$$\forall X,Y \in I_{IS}^{Pt} \quad H(I_{IS}^{Pt}) = H(1_X) \cup H(X,Y) \cup H(1_Y); \tag{3.41}$$

$H(K_{IS}^{Pt})$ – подмножество морфизмов, описывающих преобразования структурных и поведенческих паттернов представлений требований к ИС в виде знаний самих в себя и друг в друга, имеющее следующий вид:

$$\forall X,Y \in K_{IS}^{Pt} \quad H(K_{IS}^{Pt}) = H(1_X) \cup H(X,Y) \cup H(1_Y); \tag{3.42}$$

$H(D_{IS}^{Pt}, I_{IS}^{Pt})$ – подмножество морфизмов, описывающих преобразования структурных и поведенческих паттернов представлений требований к ИС в виде данных в структурные и поведенческие паттерны представлений требований к ИС в виде информации;

$H(I_{IS}^{Pt}, D_{IS}^{Pt})$ – подмножество морфизмов, описывающих преобразования структурных и поведенческих паттернов представлений требований к ИС в виде информации в структурные и поведенческие паттерны представлений требований к ИС в виде данных;

$H(D_{IS}^{Pt}, K_{IS}^{Pt})$ – подмножество морфизмов, описывающих преобразования структурных и поведенческих паттернов представлений требований к ИС в виде данных в структурные и поведенческие паттерны представлений требований к ИС в виде знаний;

$H(K_{IS}^{Pt}, D_{IS}^{Pt})$ – подмножество морфизмов, описывающих преобразования структурных и поведенческих паттернов представлений требований к ИС в виде знаний в структурные и поведенческие паттерны представлений требований к ИС в виде данных;

$H(I_{IS}^{Pt}, K_{IS}^{Pt})$ – подмножество морфизмов, описывающих преобразования представлений структурных и поведенческих паттернов требований к ИС в виде информации в структурные и поведенческие паттерны представлений требований к ИС в виде знаний;

$H(K_{IS}^{Pt}, I_{IS}^{Pt})$ – подмножество морфизмов, описывающих преобразования структурных и поведенческих паттернов представлений требований к ИС в виде знаний в структурные и поведенческие паттерны представлений требований к ИС в виде информации.

По аналогии с понятием фактор-объекта категории [114], модель L_{IS}^{Pt} будет являться фактор-категорией для рассмотренных ранее моделей (3.10), (3.14) – (3.16) и (3.20) – (3.25). Это означает, что данная категория является обобщенным описанием различных групп требований к ИС, устанавливая для этих групп единый базис знаний о требованиях к ИС, их представлениях и способах преобразования этих представлений самих в себя и друг в друга.

Связи между моделью M_{IS}^{Pt} и моделями (3.10), (3.14) – (3.16) и (3.20) – (3.25) можно описать следующим набором одноместных ковариантных функторов:

$$\begin{gathered}
\Phi_{M_{IS}^{Pt}}^{L_{IS}^{B}} : L_{IS}^{B} \to M_{IS}^{Pt}; \Phi_{M_{IS}^{Pt}}^{L_{IS}^{IB}} : L_{IS}^{IB} \to M_{IS}^{Pt}; \Phi_{M_{IS}^{Pt}}^{L_{IS}^{s}} : L_{IS}^{s} \to M_{IS}^{Pt}; \\
\Phi_{M_{IS}^{Pt}}^{L_{IS}^{f}} : L_{IS}^{f} \to M_{IS}^{Pt}; \Phi_{M_{IS}^{Pt}}^{L_{IS}^{nf}} : L_{IS}^{nf} \to M_{IS}^{Pt}; \Phi_{M_{IS}^{Pt}}^{L_{IS}^{fw}} : L_{IS}^{fw} \to M_{IS}^{Pt}; \\
\Phi_{M_{IS}^{Pt}}^{L_{IS}^{nfw}} : L_{IS}^{nfw} \to M_{IS}^{Pt}; \Phi_{L_{IS}^{B}}^{M_{IS}^{Pt}} : M_{IS}^{Pt} \to L_{IS}^{B}; \Phi_{L_{IS}^{IB}}^{M_{IS}^{Pt}} : M_{IS}^{Pt} \to L_{IS}^{IB}; \\
\Phi_{L_{IS}^{s}}^{M_{IS}^{Pt}} : M_{IS}^{Pt} \to L_{IS}^{s}; \Phi_{L_{IS}^{f}}^{M_{IS}^{Pt}} : M_{IS}^{Pt} \to L_{IS}^{f}; \Phi_{L_{IS}^{nf}}^{M_{IS}^{Pt}} : M_{IS}^{Pt} \to L_{IS}^{nf}; \\
\Phi_{L_{IS}^{fw}}^{M_{IS}^{Pt}} : M_{IS}^{Pt} \to L_{IS}^{fw}; \Phi_{L_{IS}^{nfw}}^{M_{IS}^{Pt}} : M_{IS}^{Pt} \to L_{IS}^{nfw}.
\end{gathered} \tag{3.43}$$

Здесь функторы $\Phi_{M_{IS}^{Pt}}^{L_{IS}^{B}}, \Phi_{M_{IS}^{Pt}}^{L_{IS}^{IB}}, \Phi_{M_{IS}^{Pt}}^{L_{IS}^{s}}, \Phi_{M_{IS}^{Pt}}^{L_{IS}^{f}}, \Phi_{M_{IS}^{Pt}}^{L_{IS}^{nf}}, \Phi_{M_{IS}^{Pt}}^{L_{IS}^{fw}}, \Phi_{M_{IS}^{Pt}}^{L_{IS}^{nfw}}$ описывают методы, приемы и способы формирования паттернов требований к ИС как выделения знаний на основе представлений ранее сформулированных групп требований к ИС. Функторы $\Phi_{L_{IS}^{B}}^{M_{IS}^{Pt}}, \Phi_{L_{IS}^{IB}}^{M_{IS}^{Pt}}, \Phi_{L_{IS}^{s}}^{M_{IS}^{Pt}}, \Phi_{L_{IS}^{f}}^{M_{IS}^{Pt}}, \Phi_{L_{IS}^{nf}}^{M_{IS}^{Pt}}, \Phi_{L_{IS}^{fw}}^{M_{IS}^{Pt}}, \Phi_{L_{IS}^{nfw}}^{M_{IS}^{Pt}}$ описывают методы, приемы и способы повторного использования паттернов требований к ИС в ходе формирования представлений сформулированных групп требований к ИС.

С прикладной точки зрения модель паттернов проектирования требований к ИС (3.39) определяет общую структуру фрагмента ИТ формирования и анализа требований к ИС, обеспечивающего решение задачи повторного использования требований в новых проектах ИС. Совокупность функторов (3.43) устанавливает возможные методы, приемы и способы формирования базы знаний о требованиях к ИС и использования этой базы знаний в ходе выполнения проектов создания новых ИС.

Как уже отмечалось в подразделе 3.2, формализованным описанием множества ИТ формирования и анализа требований к ИС, которые могут быть

реализованы, является модель сформулированных требований к ИС (3.26). Добавление в эту модель модели паттерна проектирования сформулированных требований к ИС (3.39) и функторов (3.43), устанавливающих связь между моделью (3.39) и другими элементами модели (3.26), позволит сформировать модель $M_{Tr_{IS}}^{Pt}$ подмножества интеллектуальных ИТ формирования и анализа требований к ИС, которые могут быть реализованы на основе концепции выявления и использования знаний о предметной области и ИС в виде паттернов. Эта модель будет иметь следующий вид:

$$M_{Tr_{IS}}^{Pt} = [\, M_{IS}^{Pt}, L_{IS}^{B}, L_{IS}^{IB}, L_{IS}^{s}, L_{IS}^{f}, L_{IS}^{nf}, L_{IS}^{fw}, L_{IS}^{nfw}, \Phi_{M_{IS}^{Pt}}^{L_{IS}^{B}}, \Phi_{M_{IS}^{Pt}}^{L_{IS}^{IB}}, \Phi_{M_{IS}^{Pt}}^{L_{IS}^{s}},$$
$$\Phi_{M_{IS}^{Pt}}^{L_{IS}^{f}}, \Phi_{M_{IS}^{Pt}}^{L_{IS}^{nf}}, \Phi_{M_{IS}^{Pt}}^{L_{IS}^{fw}}, \Phi_{M_{IS}^{Pt}}^{L_{IS}^{nfw}}, \Phi_{L_{IS}^{B}}^{M_{IS}^{Pt}}, \Phi_{L_{IS}^{IB}}^{M_{IS}^{Pt}}, \Phi_{L_{IS}^{s}}^{M_{IS}^{Pt}}, \Phi_{L_{IS}^{f}}^{M_{IS}^{Pt}}, \Phi_{L_{IS}^{nf}}^{M_{IS}^{Pt}}, \quad (3.44)$$
$$\Phi_{L_{IS}^{fw}}^{M_{IS}^{Pt}}, \Phi_{L_{IS}^{nfw}}^{M_{IS}^{Pt}}, \Phi_{L_{IS}^{IB}}^{L_{IS}^{B}}, \Phi_{L_{IS}^{s}}^{L_{IS}^{IB}}, \Phi_{L_{IS}^{f}}^{L_{IS}^{s}}, \Phi_{L_{IS}^{nf}}^{L_{IS}^{s}}, \Phi_{L_{IS}^{nf}}^{L_{IS}^{f}}, \Phi_{L_{IS}^{fw}}^{L_{IS}^{f}}, \Phi_{L_{IS}^{nfw}}^{L_{IS}^{nf}}, \Phi_{L_{IS}^{nfw}}^{L_{IS}^{fw}} \,].$$

В соответствии с приведенным в подразделе 1.1 определением понятия «информационная технология», модель (3.44) содержит выполненные в виде функторов описания методов, приемов и способов применения комплекса средств автоматизации или его отдельных элементов, используемых для следующих видов преобразований [130]:

а) преобразования представлений различных групп требований друг в друга, позволяющие сформировать функицональные и нефункциональные требования к ИТ-услугам ИС и ИТ-сервисам, реализующим эти ИТ-услуги, с учетом главной цели деятельности ИС и выдвинутой в подразделе 3.1 концепции представления требований к ИС;

б) преобразования представлений различных групп требований к ИС в представления паттернов проектирования требований к ИС, позволяющие выявлять новые знания об автоматизируемых объектах и процессах, а также знания о новых подходах к формированию требований и способов их представления;

в) преобразования представлений паттернов проектирования требований к ИС в представления различных групп требований к ИС, позволяющие использовать знания об автоматизируемых объектах и процессах, а также знания о новых подходах к формированию требований и способов их представления в ходе формирования и уточнения отдельных требований или групп требований, выдвигаемых к конкретной разрабатываемой ИС.

Применение интеллектуальной ИТ формирования и анализа требований к ИС, в основу которой положена концепция выявления и использования знаний о предметной области и ИС в виде паттернов, позволит решить задачу

повторного использования требований к ИС за счет формирования и постоянного обновления базы знаний об автоматизируемых объектах и процессах, а также знаний о новых подходах к формированию требований и способов их представления.

3.5 Выводы

Проблемы, возникающие при попытках формализованного описания требований к ИС, во многом являются следствиями ограниченного концептуального представления этих требований. Основными такими ограничениями следует считать:

- представление каталога требований к ИС как множества сформулированных требований, согласованных между Поставщиком и Потребителем;
- ориентация представлений требований к ИС на особенности конкретного проекта создания ИС, что затрудняет повторное использование этих требований в других проектах.

Для преодоления этих и других ограничений предложена концепция представления требований к ИС. Данная концепция представляет собой набор следующих положений:

а) отказ от рассмотрения только множества сформулированных требований к ИС и изначальное представление требований к ИС как элементов универсума, включающего в себя как известные, так и неизвестные Поставщику, Потребителю или им обоим требования к ИС, а также методы формирования этих требований;

б) изначальное многообразие представлений требований к ИС в виде данных, информации и знаний;

в) процессный подход к описанию требований, определяющий минимальную процессную атрибутивную модель требования к ИС;

г) подход к управлению требованиями к ИС, основанный на изложенном в подразделе 1.5 основном принципе управления требованиями.

Изложенная концепция определяет основные особенности формализованного описания невыявленных требований (3.10), частично выявленных требований (3.14) и (3.15), а также групп требований к ИС (3.16), (3.20) – (3.25). Эти описания представляют собой модели, устанавливающие связи между различными представлениями отдельных требований к ИС в пределах соответствующих множеств и групп.

На основе этих моделей были разработаны модели сформулированных требований к ИС (3.26) и универсума требований к ИС (3.27). Физический смысл данных моделей заключается в следующем:

а) модель универсума требований (3.27) является формализованным описанием фрагмента допустимых для реализации методологий

проектирования ИС, в котором рассматриваются возможные модели и методы формирования и анализа требований к ИС;

б) модель сформулированных требований (3.26) с прикладной точки зрения является формализованным описанием набора допустимых для реализации ИТ формирования и анализа требований к ИС.

Модели (3.26) и (3.27) предполагают осуществление преобразований представлений групп требований друг в друга как реализацию соответствующих одноместных ковариантных функторов. В связи с этим возникает вопрос о возможности разработки единого подхода к формализованному описанию этих функторов. Проведенное исследование позволило выделить в качестве основы такого подхода формализованное описание одноместного ковариантного функтора в виде выражения (3.38), позволяющее установить два основных способа реализации подобного функтора. Первый способ предполагает формирование и реализацию некоего общего алгоритма или базы правил преобразования исходного представления требования к ИС в конечное представление этого же требования или множества других требований. При этом предполагается, что текстовые конструкции или элементы визуальных моделей, описывающие эти требования, заранее известны алгоритму или базе правил и практически не меняются с течением времени. Второй способ предполагает создание и постоянное развитие некоего множества алгоритмов или некоей системы правил преобразования исходного представления требования к ИС в конечное представление этого же требования или множества других требований. При этом темпы изменения этого множества алгоритмов или систем правил определяются темпами появления новых или изменения существующих понятий тезауруса исходного представления требования к ИС.

Предложенное описание одноместного ковариантного функтора как основного способа преобразования категорных моделей групп требований друг в друга выводит на передний план проблему повторного использования требований к ИС, сформулированных в предыдущих проектах создания различных ИС, в ходе выполнения проекта создания новой ИС. В качестве решения этой проблемы предложено использовать концепцию паттернов проектирования, адаптировав ее положения к особенностям представления требований к ИС. Было разработано определение паттерна проектирования требования к ИС и предложена математическая модель (3.39), описывающая подобные паттерны. Опираясь на выделенное множество функторов (3.43), было предложено улучшить модель (3.26) за счет включения в нее механизмов формирования и обработки паттернов проектирования требований к ИС. В результате этого улучшения была получена модель (3.44) подмножества интеллектуальных ИТ формирования и анализа требований к ИС, которые могут быть реализованы на основе концепции выявления и использования знаний о предметной области и ИС в виде паттернов. Применение интеллектуальной ИТ формирования и анализа требований к ИС, в основу которой положена концепция выявления и использования знаний о предметной

области и ИС в виде паттернов, позволит решить задачу повторного использования требований к ИС за счет формирования и постоянного обновления базы знаний об автоматизируемых объектах и процессах, а также знаний о новых подходах к формированию требований и способов их представления. Таким образом, вместо повторного использования отдельных представлений требований к ИС предлагается осуществлять повторное использование знаний, выявленных в ходе формирования и анализа требований, выдвинутых к различным ранее разрабатывавшимся ИС.

4 ФОРМАЛИЗОВАННОЕ ОПИСАНИЕ АРХИТЕКТУРНОГО ФРЕЙМВОРКА УСКОРЕННОЙ РАЗРАБОТКИ ИНФОРМАЦИОННОЙ СИСТЕМЫ

4.1 Обобщенная модель архитектурного фреймворка информационной системы. Архитектурный фреймворк ускоренной разработки информационной системы

Изложенная в разделе 3 концепция представления требований к ИС, в частности, модель подмножества интеллектуальных ИТ формирования и анализа требований к ИС (3.44) не зависит от вида архитектуры конкретной ИС. Поэтому до начала программной реализации любой из ИТ данного подмножества следует решить задачу адаптации модели (3.44) к особенностям конкретной архитектуры ИС.

Решением этой задачи в общем случае является архитектурный фреймворк, определяющий особенности описания ИС, ее элементов и архитектуры, а также особенности решения задач интеграции ИС из отдельных элементов на различных уровнях представления. Модель такого фреймворка должна отражать тот набор методологических положений, знаний, видов моделей и методов, правил и практик, которые могут быть использованы в процессах создания ИС. Место архитектурного фреймворка в процессах создания конкретной ИС показано на рис. 4.1.

Рисунок 4.1 – Место архитектурного фреймворка в процессах создания информационной системы

Из рис. 4.1 следует, что основная точка зрения участников процессов создания ИС на архитектурный фреймворк представляет его как своеобразный фильтр, выделяющий и формирующий из всего множества концептуальных положений, видов моделей, методов, знаний, правил и практик создания ИС внутренне непротиворечивое подмножество, элементы которого необходимы для создания конкретной ИС. Под внутренней непротиворечивостью здесь следует понимать существование отображений, связывающих концептуальные положения, виды моделей, методы, знания, правила и практики создания ИС в единую целостную систему. При этом такие отображения всегда направлены «от теории к практике» – так, например, используемый в архитектурном фреймворке вид модели ИС может стать началом отображения, концом которого будет являться множество практик применения моделей данного вида в процессах создания ИС, однако обратное неверно[13].

Основываясь на данном представлении, любой архитектурный фреймворк можно разделить на две части. К первой части – формальной – относятся поддающиеся формальному описанию виды моделей, методы, декларативные знания, процедурные правила и алгоритмы, которые могут быть использованы при решении задач, возникающих в процессах создания ИС. Ко второй части – неформальной – относятся неформальные или слабо формализуемые концептуальные положения, стандарты, знания, правила и практики, которые могут быть использованы конкретными представителями Поставщика и Потребителя для успешного выполнения процессов создания ИС. Такое разделение позволяет рассматривать первую часть архитектурного фреймворка в виде своеобразного банка видов моделей, методов, знаний, правил и алгоритмов, элементы которого используются для описаний функторов (3.43) любой из возможных интеллектуальных ИТ формирования и анализа требований к ИС, описанных моделью (3.44), простейшей моделью такого банка является ориентированный граф, показанный на рис. 4.2.

Однако такое представление является излишне упрощенным. Поэтому целесообразно описать формальную часть архитектурного фреймворка в виде следующей модели:

$$M_{AF} = [\, L_{Md}, L_{Mt}, L_{Kn}, L_{Al}, \Phi_{L_{Mt}}^{L_{Md}}, \Phi_{L_{Kn}}^{L_{Md}}, \Phi_{L_{Al}}^{L_{Md}}, \Phi_{L_{Kn}}^{L_{Mt}}, \Phi_{L_{Al}}^{L_{Mt}}, \Phi_{L_{Al}}^{L_{Kn}} \,], \quad (4.1)$$

где M_{AF} – обобщенная модель формальных элементов архитектурного фреймворка;

L_{Md} – модель набора видов моделей, используемых архитектурным фреймворком;

[13] В данном случае слова «обратное неверно» характерны только для отображений в рамках самого архитектурного фреймворка. Вполне реальна ситуация, когда по результатам применения практик создания ИС (например, аналогичных изложенным в стандартах ISO), решаются задачи Data Mining или Process Mining, в результате чего формируются новые теоретические знания или модели.

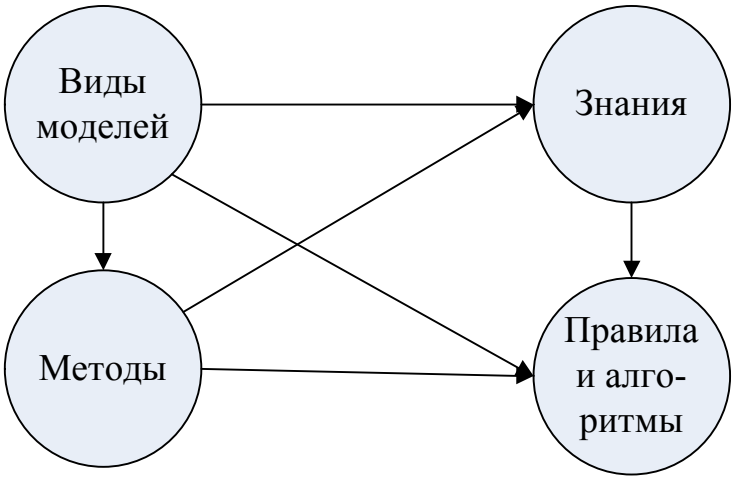

Рисунок 4.2 – Обобщенное представление формальной части архитектурного фреймворка

L_{Mt} – модель набора методов, используемых архитектурным фреймворком;

L_{Kn} – модель набора декларативных знаний, используемых архитектурным фреймворком;

L_{Al} – модель набора процедурных правил и алгоритмов, используемых архитектурным фреймворком;

$\Phi_{L_{Mt}}^{L_{Md}}$ – одноместный ковариантный функтор, устанавливающий связь между моделями L_{Md} и L_{Mt};

$\Phi_{L_{Kn}}^{L_{Md}}$ – одноместный ковариантный функтор, устанавливающий связь между моделями L_{Md} и L_{Kn};

$\Phi_{L_{Al}}^{L_{Md}}$ – одноместный ковариантный функтор, устанавливающий связь между моделями L_{Md} и L_{Al};

$\Phi_{L_{Kn}}^{L_{Mt}}$ – одноместный ковариантный функтор, устанавливающий связь между моделями L_{Mt} и L_{Kn};

$\Phi_{L_{Al}}^{L_{Mt}}$ – одноместный ковариантный функтор, устанавливающий связь между моделями L_{Mt} и L_{Al};

$\Phi_{L_{Al}}^{L_{Kn}}$ – одноместный ковариантный функтор, устанавливающий связь между моделями L_{Kn} и L_{Al}.

Из модели (4.1) следует, что основным элементом любого архитектурного фреймворка создаваемой ИС следует считать набор видов моделей, используемых конкретным архитектурным фреймворком в процессах создания ИС с соответствующей архитектурой. Поэтому в данной книге основное

внимание будет сосредоточено на детализации описаний элементов модели L_{Md} и особенностях практической реализации этих описаний. Описания элементов других категорий и функторов модели (4.1) требуют специальных исследований и здесь не рассматриваются.

Использование изложенной в разделе 3 концепции представления требований к ИС кардинально изменяет модели, методы, знания, правила и алгоритмы, используемые в ходе выполнения процессов, непосредственно работающих с требованиями, а также процесса проектирования архитектуры системы. Поэтому становится возможным говорить о существовании архитектурного фреймворка, основанного на положениях сервисного подхода и предложенной концепции представления требований к ИС, Такой архитектурный фреймворк в дальнейшем будем называть ***архитектурным фреймворком ускоренной разработки ИС*** (АФУР ИС). В соответствии с описанием понятия «архитектурный фреймворк», приведенным в подразделе 1.3, модель АФУР ИС должна установить общие особенности и ограничения создания, анализа, интерпретации и использования AD ИС на основе требований к ИС, применяемых в указанных выше процессах создания ИС.

Использование концепции представления требований к ИС, предложенной в разделе 3, позволяет уточнить взаимосвязи основных элементов понятия «описание архитектуры системы на основе системного подхода» (см. рис. 2.2). Контекстная диаграмма классов понятия «описание архитектуры системы на основе сервисного подхода с точки зрения требований к системе» показана на рис. 4.3.

Создание ИС с применением АФУР ИС, основанное на предложенной контекстной диаграмме классов, подразумевает разработку и интеграцию описаний архитектуры на трех основных уровнях представления: общесистемном уровне, уровне ИТ-услуг и уровне ИТ-сервисов. При этом процесс проектирования архитектуры системы, который может осуществляться как сверху вниз, так и снизу вверх, предполагает обязательную интеграцию описаний элементов ИС, выполненных на разных уровнях представления. Так, например, описание архитектуры ИТ-сервиса должно включать описание взаимосвязанных функциональных операций, в совокупности обеспечивающих физическую реализацию функциональности сервиса; описание архитектуры ИТ-услуги – описание ИТ-сервисов, в совокупности обеспечивающих реализацию данной ИТ-услуги; описание архитектуры ИС – описания взаимосвязанных ИТ-услуг. Другими словами, архитектура ИС, как и AD ИС, на всех уровнях представления рассматривается как набор автономных модулей, обеспечивающих реализацию конкретного варианта конфигурации ФС ИС. Каждый такой модуль любого уровня представления характеризуется набором входных данных, необходимых для получения информации извне, и выходных данных, необходимых для отображения результатов выполнения ИТ-услуги ИС, а также интерфейсов, обеспечивающих взаимодействие модуля с внешней средой. При этом внешней средой могут являться другие модули этой

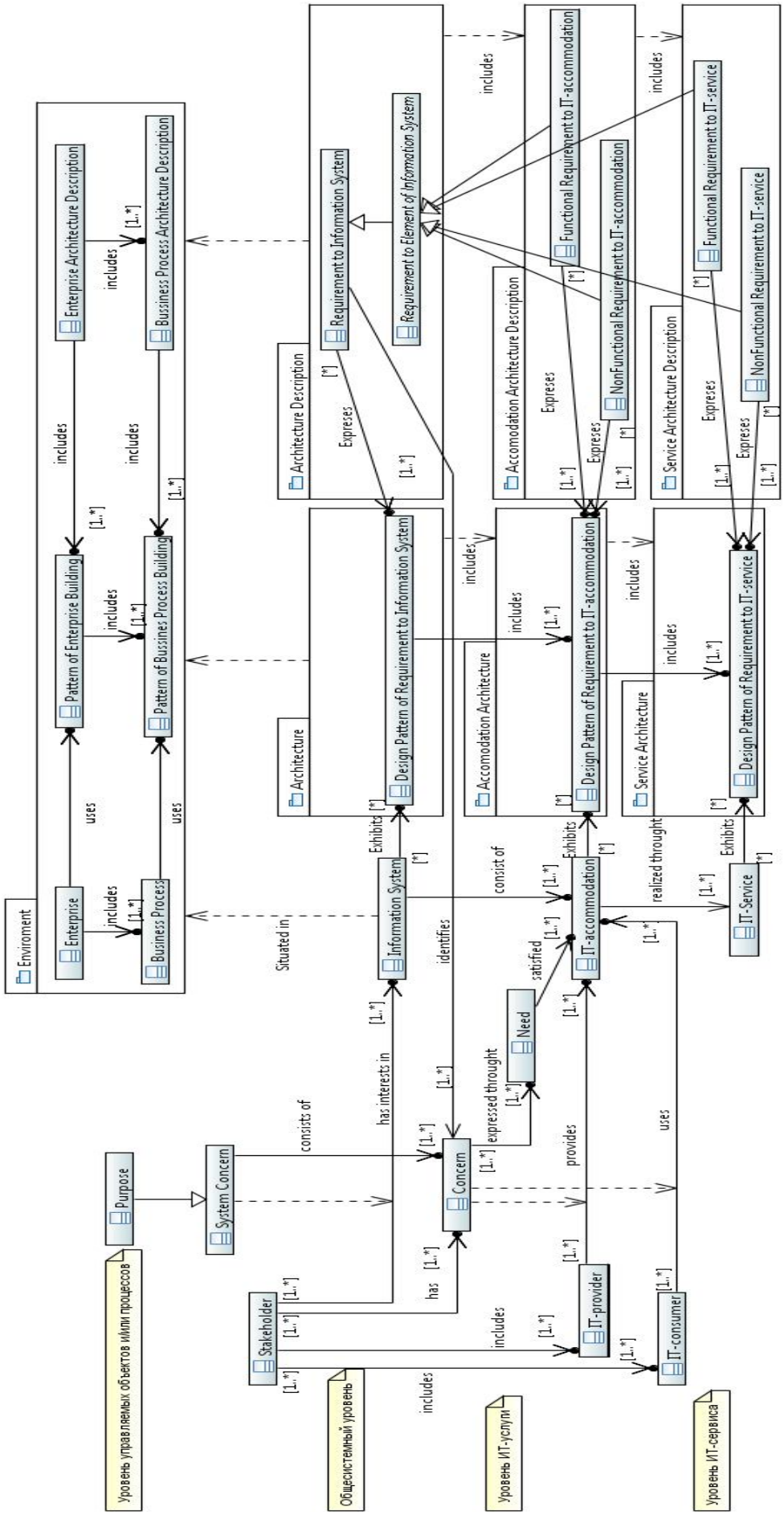

Рисунок 4.3 – Контекстная диаграмма классов понятия «описание архитектуры системы на основе сервисного подхода с точки зрения требований к системе»

же ИС, внешние системы, или пользователи ИС, взаимодействующие с модулем через пользовательский интерфейс.

Такая точка зрения на АФУР ИС определяет как наиболее предпочтительный способ проектирования архитектуры ИС методом «снизу вверх». Это обусловлено наличием библиотеки готовых элементов ИС (ИТ-услуг и отдельных ИТ-сервисов), эффект от повторного использования которых увеличивается при оперировании описанием элементов ИС, выполненным на уровне отдельных ИТ-услуг и особенно отдельных ИТ-сервисов. Данный уровень представления позволяет обеспечить повторное использование ИТ-сервисов, даже в том случае, когда не востребована полная функциональность созданных ранее ИТ-услуг, реализацию которых обеспечивают данные ИТ-сервисы. В этом случае такие ИТ-сервисы становятся элементами новой ИТ-услуги.

Поэтому процесс проектирования архитектуры системы в рамках АФУР ИС следует рассматривать как процесс синтеза варианта конфигурации ФС ИС из отдельных ИТ-услуг. При этом предполагается, что повторное использование требований к ИС позволит идентифицировать те ИТ-услуги, которые находятся в библиотеке готовых решений, доступной Поставщику в ходе процессов создания конкретной ИС. Процессы, непосредственно работающие с требованиями к ИС, в этом случае целесообразно рассматривать как процессы идентификации требований к системе в целом, функциональных и нефункциональных требований к ИТ-услугам, образующим ФС ИС.

Сказанное выше позволяет рассматривать модель АФУР ИС как частный случай модели (4.1), в которой компонент L_{Md} определяется как совокупность паттернов проектирования требований к ИС, в общем случае описываемых моделью (3.39). Тогда модель АФУР ИС будет иметь следующий вид:

$$M_{AF} = [\, M_{str}^{Pt}, M_{bhv}^{Pt}, L_{Kn}, L_{Al}, \Phi_{M_{bhv}^{Pt}}^{M_{str}^{Pt}}, \Phi_{L_{Kn}}^{M_{str}^{Pt}}, \Phi_{L_{Al}}^{M_{str}^{Pt}}, \Phi_{L_{Kn}}^{M_{bhv}^{Pt}}, \Phi_{L_{Al}}^{L_{bhv}^{Pt}}, \Phi_{L_{Al}}^{L_{Kn}} \,], \quad (4.2)$$

где M_{str}^{Pt} – модель структурных паттернов проектирования требований к ИС (см. подраздел 3.4), являющаяся частным случаем модели (3.39);

M_{bhv}^{Pt} – модель поведенческих паттернов проектирования требований к ИС, (см. подразд. 3.4), являющаяся частным случаем модели (3.39);

$\Phi_{M_{bhv}^{Pt}}^{M_{str}^{Pt}}$ – одноместный ковариантный функтор, устанавливающий связь между моделями M_{str}^{Pt} и M_{bhv}^{Pt};

$\Phi_{L_{Kn}}^{M_{str}^{Pt}}$ – одноместный ковариантный функтор, устанавливающий связь между моделями M_{str}^{Pt} и L_{Kn};

$\Phi_{L_{Al}}^{M_{str}^{Pt}}$ – одноместный ковариантный функтор, устанавливающий связь между моделями M_{str}^{Pt} и L_{Al};

$\Phi_{L_{Kn}}^{M_{bhv}^{Pt}}$ – одноместный ковариантный функтор, устанавливающий связь между моделями M_{bhv}^{Pt} и L_{Kn};

$\Phi_{L_{Al}}^{M_{bhv}^{Pt}}$ – одноместный ковариантный функтор, устанавливающий связь между моделями M_{bhv}^{Pt} и L_{Al}.

Следует отметить, что, как и для модели (4.1), основным элементом АФУР ИС следует считать набор структурных паттернов проектирования требований, устанавливающих допустимые для использования виды моделей и формализованных описаний различных групп требований к ИС. Поэтому в дальнейшем основное внимание будет уделяться решению проблемы детализации описаний моделей структурных паттернов проектирования требований к ИС, используемых АФУР ИС.

4.2 Способы представления требований к информационной системе и паттернов проектирования требований

Для успешной реализации структурных паттернов проектирования требований к ИС в рамках АФУР ИС необходимо выделить основные способы представления различных групп требований к ИС. В общем случае, как следует из изложенного выше материала, представления отдельных требований к ИС определяются следующими условиями:

а) определение главной цели деятельности ИС как формирования и отображения единого целостного информационного представления объекта или процесса в соответствии с поставленными перед системой целями;

б) сформулированная в подразделе 3.1. концепция представления требований к ИС;

в) категорные модели отдельных групп требований к ИС (3.16), (3.20) – (3.25), категорная модель сформулированных требований к ИС (3.26) и категорная модель универсума требований к ИС (3.27);

г) рассмотренные в подразделе 1.5 атрибутивные модели описания различных групп требований к ИС.

Исходя из этих условий, предлагаются следующие способы представления отдельных требований к ИС на уровне данных, информации и знаний (табл. 4.1).

Таблица 4.1 – Описание способов представлений отдельных требований к информационной системе на различных уровнях

Виды представлений требования к ИС	Способ представления требования к ИС
Представление требования к ИС на уровне данных	Атрибутивная модель требования, используемая для идентификации и управления требованием к ИС
Представление требования к ИС на уровне информации	Описание требования к ИС в виде текстов на естественном языке, множестве формальных языков (например, в виде визуальных моделей) или же в виде набора целевых показателей, характеризующих степень удовлетворения требования к ИС
Представление требования к ИС на уровне знаний	Онтология элемента управляемого объекта или процесса, элемента ИС или ИС в целом, к которым выдвигается требование

Предложенные в табл. 4.1 способы представлений отдельных требований к ИС определяют дополнительное условие, дополняющее рассмотренные выше условия для случая формирования представлений паттернов проектирования требований к ИС. Это условие заключается в следующем: различные представления одного и того же требования к ИС не должны искажать заложенную в это требование Поставщиком и Потребителем семантику автоматизируемого объекта, процесса, разрабатываемой ИС или ее элемента. Для выполнения этого условия необходимо выбрать подход к решению проблемы формирования единых описаний автоматизируемого объекта или процесса, а также разрабатываемой ИС.

На сегодняшний день имеется несколько равноправных методов решения данной проблемы. Одним из таких методов является создание специализированных описаний и формализованных представлений, которые определяют синтаксис и семантику конкретных реализаций ИС и ее компонентов [109]. Такие описания и формализованные представления М. Фаулер предложил называть метамоделями. Хотя термин «метамодель» в понимании Фаулера приобрел более узкий смысл, суть его осталась той же, что и в традиционной теории систем, где существование метаописания и метамодели является и необходимым, и конструктивным [131].

Некоторые современные исследователи-практики особое внимание обращают не на то, каким образом метамодель можно формализовать и поместить в текст, именуемый “общая теория систем”, а на то, что метамодель присутствует в конструкции систем как принцип устройства этой конструкции и одновременно как механизм, обеспечивающий конструирование, существование и взаимодействие множеств моделей [132]. При этом сам механизм реализуется не с помощью специально выделенных структур данных или индексации структур – это очевидно вытекает из факта существования Геделевской схемы нумерации. Так А.О. Поляков в [133] указывает, что все естественные открытые системы имеют не гомеостатический, а гомеокинетический характер поведения и устройства. Представляется разумным предположить, что реализация метамодели ИС как открытой системы включает в себя не только определенные типы структур данных и взаимодействие этих структур, но и определенные типы и способы организации этих взаимодействий, то есть имеет характер сложно организованной, но целостной системы взаимодействий и способов организации этих взаимодействий. Наиболее приемлемым способом организации метамодели А.О. Поляков видит реализацию некоторого механизма, обеспечивающего раздельную организацию работы механизма логического вывода и механизма интерпретации результатов вывода [133].

Использование метамоделей для описания и принципов устройства моделей представлений требований к ИС на различных уровнях и механизма, обеспечивающего конструирование, существование и взаимодействие множеств моделей представлений требований к ИС на различных уровнях, позволяет определить способы представления паттернов проектирования требований к ИС следующим образом (табл. 4.2).

Таблица 4.2 – Описание способов представления паттерна проектирования требований к информационной системе на различных уровнях

Виды представления паттерна проектирования требования к ИС	Способ представления паттерна проектирования требования к ИС
1	2
Представление паттерна проектирования требования к ИС на уровне данных	Метамодель, определяющая синтаксис и семантику построения атрибутивной модели требования, принадлежащего конкретной группе требований к ИС
Представление паттерна проектирования требования к ИС на уровне информации	Метамодель, определяющая синтаксис и семантику описаний требования, принадлежащего конкретной группе требований к ИС, в виде текстов на естественном языке или же на множестве формальных языков (например, в виде визуальных моделей)

Продолжение табл. 4.2

1	2
Представление паттерна проектирования требования к ИС на уровне знаний	Описание базовых онтологий предметной области или ИС, а также метамодель, определяющая синтаксис и семантику онтологии требования, принадлежащего конкретной группе требований к ИС, на основе выделенных базовых онтологий

4.3 Разработка структурных паттернов проектирования требований к информационной системе на уровне данных

Как показано в подразделе 4.2, представлением требования к ИС на уровне данных предложено считать атрибутивную модель этого требования. Данная модель применяется для выполнения операции хранения требования в каталоге требований к ИС, а также для управления этим требованием. Представлением паттерна проектирования требования к ИС на уровне данных предложено считать метамодель, определяющую синтаксис и семантику построения атрибутивной модели требования, принадлежащего конкретной группе требований к ИС.

Однако, как показано в подразделе 1.5, атрибутивные модели требований к ИС могут серьезно различаться при описании требований, принадлежащих к разным группам требований к ИС. Кроме того, не существует научно обоснованных гарантий адекватности каких бы то ни было атрибутивных моделей требований процессам разработки ИС.

В то же время можно выделить целый ряд групп показателей, которые необходимы для выполнения таких стандартных формальных операций, как ввод нового требования, отображение введенного требования, редактирование ранее введенного требования, удаление ранее введенного требования, а также для осуществления операций управления требованием к ИС. Эти показатели должны присутствовать во всех атрибутивных моделях требований всех групп требований к ИС.

В дополнение к сказанному следует отметить, что разрабатываемая модель представления требований на уровне данных и метамодель, определяющая ее синтаксис и семантику, должны быть в максимальной степени независимы от особенностей их технологической реализации. В частности, эти модель и метамодель должны быть в максимальной степени независимы от особенностей формального описания конкретных моделей данных, которые, в свою очередь, определяют особенности выполнения операций хранения и управления представлениями требований к ИС. Это условие позволит обеспечить

возможность использования разрабатываемой модели и метамодели даже в случае кардинального изменения ИТ хранения данных.

Для разработки модели представления требований к ИС на уровне данных введем по аналогии с моделями отдельных групп требований (3.16), (3.20) – (3.25) обобщенную модель сформулированного требования к ИС M_{IS}^{tr} следующего вида:

$$M_{IS}^{tr} = [\, D_{IS}^{tr}, I_{IS}^{tr}, K_{IS}^{tr}, H(D_{IS}^{tr}), H(I_{IS}^{tr}), H(K_{IS}^{tr}), H(D_{IS}^{tr}, I_{IS}^{tr}), \\ H(I_{IS}^{tr}, D_{IS}^{tr}), H(D_{IS}^{tr}, K_{IS}^{tr}), H(K_{IS}^{tr}, D_{IS}^{tr}), H(I_{IS}^{tr}, K_{IS}^{tr}), H(K_{IS}^{tr}, I_{IS}^{tr})], \quad (4.3)$$

где D_{IS}^{tr} – подкласс представлений сформулированных требований к ИС на уровне данных;

I_{IS}^{tr} – подкласс представлений сформулированных требований к ИС на уровне информации;

K_{IS}^{tr} – подкласс представлений сформулированных требований к ИС на уровне знаний;

$H(D_{IS}^{tr})$ – подмножество морфизмов, описывающих преобразования представлений сформулированных требований к ИС на уровне данных самих в себя и друг в друга, имеющее следующий вид:

$$\forall X, Y \in D_{IS}^{tr} \quad H(D_{IS}^{tr}) = H(1_X) \cup H(X, Y) \cup H(1_Y); \quad (4.4)$$

$1_X, 1_Y$ – единичные морфизмы, условия существования которых описаны в [114];

$H(I_{IS}^{tr})$ – подмножество морфизмов, описывающих преобразования представлений сформулированных требований к ИС на уровне информации самих в себя и друг в друга, имеющее следующий вид:

$$\forall X, Y \in I_{IS}^{tr} \quad H(I_{IS}^{tr}) = H(1_X) \cup H(X, Y) \cup H(1_Y); \quad (4.5)$$

$H(K_{IS}^{tr})$ – подмножество морфизмов, описывающих преобразования представлений сформулированных требований к ИС на уровне знаний самих в себя и друг в друга, имеющее следующий вид:

$$\forall X, Y \in K_{IS}^{tr} \quad H(K_{IS}^{tr}) = H(1_X) \cup H(X, Y) \cup H(1_Y); \quad (4.6)$$

$H(D_{IS}^{tr}, I_{IS}^{tr})$ – подмножество морфизмов, описывающих методы, приемы и способы преобразования представлений сформулированных

требований к ИС на уровне данных в представления сформулированных требований к ИС на уровне информации;

$H(I_{IS}^{tr}, D_{IS}^{tr})$ – подмножество морфизмов, описывающих методы, приемы и способы преобразования представлений сформулированных требований к ИС на уровне информации в представления сформулированных требований к ИС на уровне данных;

$H(D_{IS}^{tr}, K_{IS}^{tr})$ – подмножество морфизмов, описывающих методы, приемы и способы преобразования представлений сформулированных требований к ИС на уровне данных в представления сформулированных требований к ИС на уровне знаний;

$H(K_{IS}^{tr}, D_{IS}^{tr})$ – подмножество морфизмов, описывающих методы, приемы и способы преобразования представлений сформулированных требований к ИС на уровне знаний в представления сформулированных требований к ИС на уровне данных;

$H(I_{IS}^{tr}, K_{IS}^{tr})$ – подмножество морфизмов, описывающих методы, приемы и способы преобразования представлений сформулированных требований к ИС на уровне информации в представления сформулированных требований к ИС на уровне знаний;

$H(K_{IS}^{tr}, I_{IS}^{tr})$ – подмножество морфизмов, описывающих методы, приемы и способы преобразования представлений сформулированных требований к ИС на уровне знаний в представления сформулированных требований к ИС на уровне информации.

Данную модель целесообразно использовать для выделения тех элементов моделей представлений требований к ИС на различных уровнях, которые должны быть едиными для различных групп требований и определяются паттернами проектирования требования к ИС.

Тогда модель представления сформулированных требований к ИС на уровне данных следует представлять как формализованное описание каждого элемента подкласса D_{IS}^{tr} с последующим уточнением этого описания после отнесения данного элемента к одному из следующих подклассов: D_{IS}^{B}, D_{IS}^{IB}, D_{IS}^{s}, D_{IS}^{f}, D_{IS}^{nf}, D_{IS}^{fw}, D_{IS}^{nfw}. Основываясь на этой точке зрения, модель подкласса представлений сформулированных требований к ИС на уровне данных будет представлять собой кортеж атрибутов, который структурно разделен на следующие части:

$$M_{D_{IS}^{tr}} = \left\langle < M_{D}^{At_{Pt}} >, < M_{D}^{At_{gr}} > \right\rangle, \tag{4.7}$$

где $M_{D_{IS}^{tr}}$ – модель подкласса представлений сформулированных требований к ИС на уровне данных;

$< M_D^{At_{Pt}} >$ – кортеж элементов атрибутивной модели требования к ИС, которые определяются паттернами проектирования требований к ИС на уровне данных и являются обязательными для требований любой группы;

$< M_D^{At_{gr}} >$ – кортеж элементов атрибутивной модели требования к ИС, которые определяются, исходя из индивидуальных особенностей выполнения Поставщиком и Потребителем процессов, непосредственно работающих с представлениями требований на уровне данных конкретной группы.

В общем случае конкретный вид элемента $< M_D^{At_{Pt}} >$ модели (4.7) определяется совокупностью структурных и поведенческих паттернов проектирования требований к ИС. Однако, как было показано выше, поведенческие паттерны определяются особенностями структурных паттернов проектирования требований к ИС. На практике это означает, что конкретные методы и алгоритмы выполнения основных операций над требованиями к ИС на уровне данных могут быть выделены из множества общих методов на основе как общей специфики процессов создания ИС, так и конкретных особенностей структурных паттернов проектирования требований к ИС. Поэтому основное внимание в данном подразделе уделяется разработке структурных паттернов проектирования требований к ИС, определяющих семантику общих представлений требований различных групп на уровне данных в ходе выполнения основных операций над требованиями. В качестве таких операций следует выделить:

а) идентификацию требования к ИС;

б) управление требованием к ИС;

в) операции создания, отображения, редактирования и удаления описаний отдельных требований, версий этих требований, а также представлений требований и их версий.

Рассмотрим структурный паттерн проектирования требований к ИС на уровне данных для операции идентификации требований различных групп. В данном случае под термином «идентификация требования» понимается установление следующих характеристик требования к системе (см. табл. А.1 Приложения А) :

а) уникальное описание каждого требования;

б) описание взаимосвязей между требованиями и правообладателями;

в) уникальное описание правообладателей требований;

г) описание взаимосвязей требований друг с другом для прослеживания каждого требования до первоначальных требований правообладателей.

В дополнение к данным характеристикам при идентификации требований к ИС, которые могут относиться к разным группам требований, необходимо установление таких характеристик:

а) уникальное описание каждой группы требований к ИС;

б) уникальное описание каждой версии требования к ИС;

в) уникальное описание сотрудников Поставщика ИТ-услуг, которые либо являются авторами требования к ИС или версии этого требования, либо выполняют работу по формулированию требования к ИС или версии этого требования, высказанного представителем (или представителями) Потребителя ИТ-услуг;

г) уникальное описание проекта создания ИС, в ходе которого было сформулировано требование к ИС или версия этого требования;

д) факт использования требования к ИС или версии этого требования в проекте создания ИС.

При этом стоит учесть, что для повторного использования требований к ИС или версий этих требований в проектах создания различных ИС необходимо, чтобы описание требования к ИС было первичным по отношению к описанию проекта создания конкретной ИС.

На основании этих показателей предлагается структурный паттерн идентификации требования к ИС, который имеет следующий вид:

$$Pt_{Id} =< At_{rg}, At_{r}, At_{asp}, At_{acs}, At_{proj}, < at_{rg}, at_{r}, at_{asp}, at_{acs}, at_{proj} >>, \quad (4.8)$$

где Pt_{Id} – модель структурного паттерна идентификации требования к ИС, определяющего семантику моделей требований к ИС различных групп;

At_{rg} – кортеж атрибутов, описывающих группы требований к ИС;

At_{r} – кортеж атрибутов, описывающих отдельные сформулированные требования к ИС;

At_{asp} – кортеж атрибутов, описывающих сотрудников Поставщика как возможных авторов требований к ИС;

At_{acs} – кортеж атрибутов, описывающих сотрудников Потребителя как возможные источники требований к ИС;

At_{proj} – кортеж атрибутов, описывающих проекты создания ИС, для которых формулируются требования;

at_{rg} – атрибут, идентифицирующий группу требований к ИС;

at_{r} – атрибут, идентифицирующий отдельное требование к ИС;

at_{asp} – атрибут, идентифицирующий сотрудника Поставщика как автора требований к ИС;

at_{acs} – атрибут, идентифицирующий сотрудника Потребителя как автора требований к ИС;

at_{proj} – атрибут, идентифицирующий проект создания ИС, для которого было сформулировано требование;

$< at_{rg}, at_r, (at_{apr}), (at_{au}), at_{proj} >$ – кортеж атрибутов, описывающий факт идентификации конкретного требования к ИС, сформулированного конкретными авторами в проекте создания конкретной ИС.

Интерпретацией модели (4.8) в ходе разработки ИТ формирования и анализа требований к ИС является схема одной или нескольких витрин данных, предназначенных для хранения следующей исторической информации:

а) о каталоге сформулированных требований, допустимых для повторного использования в проектах создания различных ИС;

б) о каталоге сформулированных требований, выдвинутых в ходе выполнения проекта создания конкретной ИС.

В случае использования модели (4.8) для идентификации версии требования к ИС, данная модель принимает следующий вид:

$$Pt_{Id} =< At_{rg}, At_r, At_{rv}, At_{asp}, At_{acs}, At_{proj}, < at_{rg}, at_r, at_{rv}, at_{asp}, at_{acs}, at_{proj} >>, \quad (4.9)$$

где At_{rv} – кортеж атрибутов, описывающих версии требования к ИС;

at_{rv} – атрибут, идентифицирующий версию требования к ИС.

Интерпретация модели (4.9) в ходе разработки ИТ формирования и анализа требований к ИС аналогична интерпретации модели (4.8).

Как было показано в подразделе 1.6, управление отдельными требованиями основано на постепенном преобразовании множества начальных значений атрибутов, описывающих требование, в множество желаемых значений тех же атрибутов. Под желаемым значением следует понимать значение, которое приобретает атрибут при описании реализованного требования, проверенного соответствующими тестами.

Вместо анализа совокупности атрибутов, определяющих состояние требования к ИС, часто предлагается использовать анализ статуса требования (см. подраздел 1.6). Такой подход позволяет организовать управление требованиями к ИС как анализ, проводимый лицами, принимающими решение (среди которых могут быть как представители Поставщика, так и представители Потребителя). Анализу подвергаются возможности изменения статуса требования и, в случае принятия положительного решения, изменения текущего значения статуса анализируемого требования.

Описанная процедура определяет описание процесса управления требованием к ИС, выполняемого в течение всего жизненного цикла требования, следующим образом (табл. 4.3).

Предложенное описание процесса управления требованиями, выполненное в соответствии с процессной моделью, рассмотренной в подразделе 3.1, определяет необходимость установления следующих характеристик:

Таблица 4.3 – Описание процесса управления требованием к информационной системе

Номер	Элемент процессной модели	Значение элемента процессной модели
1	Название (наименование) процесса	Процесс управления требованиями к ИС
2	Ожидаемые результаты выполнения процесса	Установление нового текущего статуса требования к ИС
3	Виды деятельности, выполняемые в рамках процесса	Анализ текущего статуса требования к ИС, принятие решения об изменении текущего статуса требования к ИС, изменение текущего статуса требования к ИС
4	Цели, достижение которых свидетельствует о выполнении процесса	Изменение значения текущего статуса требования к ИС на значение, желательное для Поставщика и Потребителя
5	Ресурсы, обрабатываемые процессом	Требования к ИС
6	Механизмы процесса	Лица, принимающие решение со стороны Поставщика и со стороны Потребителя
7	Управляющие воздействия процесса	Множество состояний требования к ИС, элементы которого могут назначаться текущими статусами требования к ИС

а) уникальное описание управляемого требования;

б) описание текущего статуса управляемого требования;

в) уникальное описание лиц, принимающих решение об управлении требованием со стороны Поставщика;

г) уникальное описание лиц, принимающих решение об управлении требованием со стороны Потребителя;

д) дата и время принятия управленческого решения об изменении текущего статуса требования;

е) описание причины принятия управленческого решения об изменении статуса требования;

ж) дата и время изменения текущего статуса требования.

В случае, если необходимо осуществлять управление отдельными версиями требования, учету подлежат значения следующих показателей:

а) уникальное описание управляемого требования;

б) уникальное описание управляемой версии требования;

в) описание текущего статуса управляемой версии требования;

г) уникальное описание лиц, принимающих решение об управлении версией требования со стороны Поставщика;

д) уникальное описание лиц, принимающих решение об управлении версией требования со стороны Потребителя;

е) дата и время принятия управленческого решения об изменении текущего статуса версии требования;

ж) описание причины принятия управленческого решения об изменении статуса версии требования;

и) дата и время изменения текущего статуса версии требования.

На основании этих показателей предлагается структурный паттерн проектирования требований к ИС на уровне данных для управления требованием, который имеет следующий вид:

$$Pt_{Ctrl} =< At_r, At_{cst}, At_{dsp}, At_{dcs}, < at_r, at_{cst}, at_{dsp}, at_{dcs}, at_{ddt}, at_{cause}, at_{cdt} >>, \quad (4.10)$$

где Pt_{Ctrl} – модель структурного паттерна управления требованием к ИС, определяющего семантику моделей требований к ИС различных групп;

At_{cst} – кортеж атрибутов, описывающих все возможные статусы требования к ИС;

At_{dsp} – кортеж атрибутов, описывающих лиц, принимающих решение об изменении текущего статуса требования к ИС со стороны Поставщика;

At_{dcs} – кортеж атрибутов, описывающих лиц, принимающих решение об изменении текущего статуса требования к ИС со стороны Потребителя;

at_{cst} – атрибут, идентифицирующий текущее состояние требования к ИС;

at_{dsp} – атрибут, идентифицирующий лицо, принимающее решение об изменении текущего статуса требования к ИС со стороны Поставщика;

at_{dcs} – атрибут, идентифицирующий лицо, принимающее решение об изменении текущего статуса требования к ИС со стороны Потребителя;

at_{ddt} – атрибут, описывающий дату принятия решения об изменении текущего статуса требования к ИС;

at_{cause} – атрибут, описывающий причину изменения текущего статуса требования к ИС;

at_{cdt} – атрибут, описывающий дату перевода требования к ИС в текущее состояние;

$< at_r, at_{cst}, at_{dsp}, at_{dcs}, at_{ddt}, at_{cause}, at_{cdt} >$ – кортеж атрибутов, устанавливающий факт управляющего воздействия на требование к ИС.

Интерпретацией модели (4.10) в ходе разработки ИТ формирования и анализа требований к ИС является схема одной или нескольких витрин данных, предназначенных для хранения исторической информации о целенаправленных

изменениях текущих статусов требований к ИС, в результате которых каждое требование либо выполняется с желаемой степенью удовлетворения, либо отклоняется и удаляется из дальнейшего рассмотрения в процессе проектирования архитектуры ИС.

В случае использования модели (4.10) для описания процесса управления версией требования к ИС данная модель уточняется путем добавления кортежа атрибутов At_{rv}, идентифицирующих управляемую версию требования к ИС. В результате этого уточнения модель (4.11) будет иметь следующий вид:

$$Pt_{Ctrl} =< At_r, At_{rv}, At_{cst}, At_{dsp}, At_{dcs}, < at_r, at_{rv}, at_{cst}, at_{dsp}, at_{dcs}, at_{ddt}, at_{cause}, at_{cdt} >>. \quad (4.11)$$

Интерпретация модели (4.11) в ходе разработки ИТ формирования и анализа требований к ИС аналогична интерпретации модели (4.10).

В ходе выполнения работ по формулированию требований к ИС, результатом которых является полная совокупность выявленных требований к системе, подвергаемая в дальнейшем анализу (см. табл. А.1 Приложения А), возникает необходимость в выполнении операций создания, отображения, редактирования и удаления описаний отдельных требований, версий этих требований, а также представлений требований и их версий. Выполнение этих операций может вызвать такие последствия:

а) изменение результата идентификации отдельного требования к ИС или его версии;

б) инициация процесса управления требованием к ИС или его версией как изменения текущего статуса этого требования или его версии.

Поскольку эти последствия могут иметь необратимый характер и влияют не только на отдельный проект создания конкретной ИС, но и на последующие проекты создания различных ИС, необходимо использовать фрагмент атрибутивной модели требования к ИС для учета операций, выполняемых над требованием.

Для учета фактов выполнения операций над требованиями к ИС необходимо установление таких характеристик:

а) уникальное описание операций, выполняемых над отдельным требованием;

б) уникальное описание сотрудников Поставщика, участвующих в выполнении конкретной операции над отдельным требованием к ИС;

в) уникальное описание сотрудников Потребителя, участвующих в выполнении конкретной операции над отдельным требованием к ИС;

г) дата и время выполнения операции над требованием к ИС.

Тогда с учетом результатов разработки моделей паттернов (4.8) – (4.11) предлагается модель структурного паттерна проектирования требований к ИС

на уровне данных для учета результатов выполнения операций над требованиями к ИС, которая имеет следующий вид:

$$Pt_{Op} =< At_r, At_{opr}, At_{aopsp}, At_{aopcs}, < at_r, at_{opr}, at_{aopsp}, at_{aopcs}, at_{dopr} >>, \quad (4.12)$$

где Pt_{Op} – модель структурного паттерна учета результатов выполнения операций над требованием к ИС, определяющего семантику моделей требований к ИС различных групп;

At_{opr} – кортеж атрибутов, описывающих операции, выполняемые над требованием к ИС;

At_{aopsp} – кортеж атрибутов, описывающих сотрудников Поставщика, участвующих в выполнении операции над требованием к ИС;

At_{aopcs} – кортеж атрибутов, описывающих сотрудников Потребителя, участвующих в выполнении операции над требованием к ИС;

at_{opr} – атрибут, идентифицирующий операцию, выполняемую над требованием к ИС;

at_{aopsp} – атрибут, идентифицирующий сотрудника Поставщика, участвующего в выполнении операции над требованием к ИС;

at_{aopcs} – атрибут, идентифицирующий сотрудника Потребителя, участвующего в выполнении операции над требованием к ИС;

at_{dopr} – атрибут, описывающий дату и время выполнения операции над требованием к ИС;

$< at_r, at_{opr}, at_{aopsp}, at_{aopcs}, at_{dopr} >$ – кортеж атрибутов, описывающий факт выполнения конкретной операции над конкретным сформулированным требованием к ИС.

Интерпретацией модели (4.12) в ходе разработки ИТ формирования и анализа требований к ИС является схема одной или нескольких витрин данных, предназначенных для хранения исторической информации о последовательности выполненных операций над сформулированными требованиями к ИС.

В случае использования модели (4.12) для учета результатов выполнения операций над версиями требований к ИС, данная модель будет иметь следующий вид:

$$Pt_{Op} =< At_r, At_{rv}, At_{opr}, At_{aopsp}, At_{aopcs}, < at_r, at_{rv}, at_{opr}, at_{aopsp}, at_{aopcs}, at_{doprv} >>, \quad (4.13)$$

где at_{doprv} – атрибут, описывающий дату и время выполнения операции над версией требования к ИС.

Интерпретация модели (4.13) в ходе разработки ИТ формирования и анализа требований к ИС аналогична интерпретации модели (4.12).

Таким образом, для случая использования в качестве элементарных описаний создаваемой системы отдельных требований к ИС подкласс объектов D_{IS}^{Pt} модели (3.39) следует рассматривать как множество структурных паттернов проектирования требований к ИС на уровне данных, устанавливающих конкретный вид элемента $< M_D^{At_{Pt}} >$ модели (4.7). Это множество паттернов будет в общем случае иметь следующий вид:

$$\begin{gathered} D_{IS}^{Pt} = \{ Pt_{Id}, Pt_{Ctrl}, Pt_{Op} \} = \{ < At_{rg}, At_r, At_{asp}, At_{acs}, At_{proj}, \\ < at_{rg}, at_r, at_{asp}, at_{acs}, at_{proj} >>, < At_r, At_{cst}, At_{dsp}, At_{dcs}, \\ < at_r, at_{cst}, at_{dsp}, at_{dcs}, at_{ddt}, at_{cause}, at_{cdt} >>, \\ < At_r, At_{opr}, At_{aopsp}, At_{aopcs}, < at_r, at_{opr}, at_{aopsp}, at_{aopcs}, at_{dopr} >> \}. \end{gathered} \tag{4.14}$$

Для случая использования отдельных версий требований к ИС в качестве элементарных описаний создаваемой системы подкласс объектов D_{IS}^{Pt} модели (3.39) следует рассматривать как множество структурных паттернов проектирования версий требований к ИС, устанавливающих конкретный вид элемента $< M_D^{At_{Pt}} >$ модели (4.7). Это множество паттернов будет в общем случае иметь следующий вид:

$$\begin{gathered} D_{IS}^{Pt} = \{ Pt_{Id}, Pt_{Ctrl}, Pt_{Op} \} = \{ At_{rg}, At_r, At_{rv}, At_{asp}, At_{acs}, At_{proj}, \\ < at_{rg}, at_r, at_{rv}, (at_{apr}), (at_{au}), at_{proj} >>, < At_r, At_{rv}, At_{cst}, At_{dsp}, At_{dcs}, \\ < at_r, at_{rv}, at_{cst}, at_{dsp}, at_{dcs}, at_{ddt}, at_{cause}, at_{cdt} >>, < At_r, At_{rv}, At_{opr}, \\ At_{aopsp}, At_{aopcs}, < at_r, at_{rv}, at_{opr}, at_{aopsp}, at_{aopcs}, at_{doprv} >> \}. \end{gathered} \tag{4.15}$$

Подкласс морфизмов $H(D_{IS}^{Pt})$ представляет собой совокупность моделей преобразований структурных и поведенческих паттернов представлений требований к ИС на уровне данных самих в себя и друг в друга. В соответствии с выражением (3.40), данный подкласс по отношению к подклассу объектов D_{IS}^{Pt}, представленному выражением (4.14), состоит из следующих видов морфизмов:

а) единичные морфизмы $1_{Pt_{Id}}$, $1_{Pt_{Ctrl}}$, $1_{Pt_{Op}}$, отображающие структурные паттерны Pt_{Id}, Pt_{Ctrl} и Pt_{Op} соответственно сами в себя;

б) морфизмы $H(Pt_{Id}, Pt_{Ctrl})$, $H(Pt_{Id}, Pt_{Op})$, $H(Pt_{Ctrl}, Pt_{Op})$, $H(Pt_{Op}, Pt_{Ctrl})$, $H(Pt_{Op}, Pt_{Id})$ и $H(Pt_{Ctrl}, Pt_{Id})$, устанавливающие связи между структурными паттернами Pt_{Id}, Pt_{Ctrl} и Pt_{Op}.

Единичные морфизмы структурных паттернов проектирования требований к ИС на уровне данных описывают поведенческий паттерн модификации. Суть данного поведенческого паттерна заключается в выполнении операций следующего вида:

а) добавление новых атрибутов в модель существующего структурного паттерна проектирования требований к ИС;

б) изменение существующих описаний атрибутов в модели существующего структурного паттерна проектирования требований к ИС;

в) удаление неиспользуемых атрибутов из модели существующего структурного паттерна проектирования требований к ИС.

Исходя из сказанного, существование единичного морфизма $1^{add}_{Pt_{Id}} \in 1_{Pt_{Id}}$, описывающего операцию добавления новых атрибутов в модель паттерна Pt_{Id}, определяется следующим условием:

$$\forall At_x \in AT^* \quad \exists 1^{add}_{Pt_{Id}} \in 1_{Pt_{Id}} : Pt_{Id} \to Pt_{Id} \bigcup At_x, \tag{4.16}$$

где At_x – подмножество атрибутов, по которым принято решение о целесообразности их добавления в модель паттерна Pt_{Id};

AT^* – множество атрибутов, не нарушающих целостность модели паттерна Pt_{Id};

$1^{add}_{Pt_{Id}}$ – единичный морфизм, описывающий операцию добавления атрибутов подмножества At_x в модель паттерна Pt_{Id};

$1_{Pt_{Id}}$ – подкласс единичных морфизмов модели паттерна Pt_{Id}.

Совокупность условий проверки целостности моделей структурных паттернов в общем случае определяется моделями представления знаний, на основе которых формируются модели паттернов, и здесь не рассматривается.

Тогда любой единичный морфизм $1^{add}_{Pt_{Id}}$, реализующий поведенческий паттерн модификации модели паттерна Pt_{Id}, может быть описан операцией следующего вида:

$$1^{add}_{Pt_{Id}} : [\, At_{rg} \oplus At_x \,] \oplus [\, At_r \oplus At_x \,] \oplus [\, At_{asp} \oplus at_x \,] \oplus \\ \oplus [\, At_{acs} \oplus At_x \,] \oplus [\, At_{proj} \oplus At_x \,] \oplus [\, \varnothing \oplus At_x \,], \tag{4.17}$$

где $\varnothing$ – пустое множество, представляющее собой не существовавший ранее кортеж атрибутов, который описывает новый аспект идентифицируемого требования.

Использование операции «сложение по модулю 2» позволяет отказаться от добавления тех атрибутов, которые по каким-либо причинам ранее уже были включены в кортежные модели соответствующих аспектов требований к ИС. В то же время данная операция может использоваться для включения в кортежную модель какого-либо аспекта требования к ИС атрибутов других кортежных моделей, что позволяет установить взаимосвязи между отдельными аспектами требования.

Для формализованного описания единичного морфизма $1^{upd}_{Pt_{Id}} \in 1_{Pt_{Id}}$, описывающего операцию изменения существующего описания атрибута модели паттерна Pt_{Id}, введем предположение о способе описания подобных атрибутов. Согласно этому предположению, любой атрибут любого структурного или поведенческого паттерна может быть описан выражением следующего вида [134]:

$$at =< n_{at}, T_{at}, D_{at} >, \tag{4.18}$$

где at – обозначение любого возможного атрибута, используемого в моделях любых возможных структурных и поведенческих паттернов;

n_{at} – имя атрибута at;

T_{at} – тип атрибута at;

D_{at} – домен (множество допустимых значений) атрибута at.

Данное выражение является минимально необходимой моделью атрибутов, используемых в моделях паттернов, и может быть расширено при необходимости более детального описания понятия «атрибут».

Любое изменение атрибута $at_x \in At_x$ модели структурного паттерна приведет к появлению в этой модели атрибута $at'_x \in At'_x$, для которого выполняется следующее условие:

$$\forall at_x, \forall at'_x \quad at_x \setminus at'_x \neq \varnothing. \tag{4.19}$$

Тогда существование единичного морфизма $1^{upd}_{Pt_{Id}} \in 1_{Pt_{Id}}$, описывающего операцию изменения существующего описания атрибута модели паттерна Pt_{Id}, определяется условием следующего вида:

$$\forall At_x, \forall At'_x \in AT^* \quad \exists 1^{upd}_{Pt_{Id}} \in 1_{Pt_{Id}} : Pt_{Id} \to Pt'_{Id}, \\ At_x \in Pt_{Id}, At_x \notin Pt'_{Id}, At'_x \notin Pt_{Id}, At'_x \in Pt'_{Id}. \tag{4.20}$$

Любой единичный морфизм $1^{upd}_{Pt_{Id}}$, реализующий поведенческий паттерн модификации модели паттерна Pt_{Id}, может быть описан операцией следующего вида:

$$1^{upd}_{Pt_{Id}} : [(At_{rg} \setminus At_x) \cup At'_x] \oplus [(At_r \setminus At_x) \cup At'_x] \oplus \\ \oplus [(At_{asp} \setminus At_x) \cup At'_x] \oplus [(At_{acs} \setminus At_x) \cup At'_x] \oplus [(At_{proj} \setminus At_x) \cup At'_x]. \tag{4.21}$$

Существование единичного морфизма $1^{del}_{Pt_{Id}} \in 1_{Pt_{Id}}$, описывающего операцию удаления неиспользуемых атрибутов из модели паттерна Pt_{Id} определяется условием

$$\forall At_x \in AT^* \quad \exists 1^{del}_{Pt_{Id}} \in 1_{Pt_{Id}} : Pt_{Id} \to Pt''_{Id}, At_x \in Pt_{Id}, At_x \notin Pt''_{Id}, \tag{4.22}$$

где Pt''_{Id} – модель структурного паттерна идентификации требования к ИС, не содержащая подмножество атрибутов At_x, введенных, но не используемых в модели Pt_{Id}.

Тогда любой единичный морфизм $1^{del}_{Pt_{Id}}$, реализующий поведенческий паттерн модификации модели паттерна Pt_{Id}, может быть описан операцией следующего вида:

$$1^{del}_{Pt_{Id}} : [At_{rg} \setminus At_x] \oplus [At_r \setminus At_x] \oplus [At_{asp} \setminus At_x] \oplus [At_{acs} \setminus At_x] \oplus \\ \oplus [At_{proj} \setminus At_x]. \tag{4.23}$$

Для модели паттерна Pt_{Ctrl} условия существования единичных морфизмов аналогичны условиям (4.16), (4.20) и (4.22), а модели этих морфизмов будут иметь, соответственно, следующий вид:

$$1^{add}_{Pt_{Ctrl}} : [At_{cst} \oplus At_x] \oplus [At_{dsp} \oplus At_x] \oplus [At_{dcs} \oplus At_x] \oplus [\varnothing \oplus At_x], \tag{4.24}$$

$$1^{upd}_{Pt_{Ctrl}} : [(At_{cst} \setminus At_x) \cup At'_x] \oplus [(At_{dsp} \setminus At_x) \cup At'_x] \oplus \oplus [(At_{dcs} \setminus At_x) \cup At'_x], \quad (4.25)$$

$$1^{del}_{Pt_{Ctrl}} : [At_{cst} \setminus At_x] \oplus [At_{dsp} \setminus At_x] \oplus [At_{dcs} \setminus At_x]. \quad (4.26)$$

Для модели паттерна Pt_{Op} условия существования единичных морфизмов аналогичны условиям (4.16), (4.20) и (4.22), а модели этих морфизмов будут иметь, соответственно, следующий вид:

$$1^{add}_{Pt_{Op}} : [At_{opr} \oplus At_x] \oplus [At_{aopsp} \oplus At_x] \oplus [At_{aopcs} \oplus At_x] \oplus \oplus [\varnothing \oplus At_x], \quad (4.27)$$

$$1^{upd}_{Pt_{Op}} : [(At_{opr} \setminus At_x) \cup At'_x] \oplus [(At_{aopsp} \setminus At_x) \cup At'_x] \oplus \oplus [(At_{aopcs} \setminus At_x) \cup At'_x], \quad (4.28)$$

$$1^{del}_{Pt_{Op}} : [At_{opr} \setminus At_x] \oplus [At_{aopsp} \setminus At_x] \oplus [At_{aopcs} \setminus At_x]. \quad (4.29)$$

Морфизмы, устанавливающие связи между моделями структурных паттернов, реализуют поведенческий паттерн интеграции описаний. Суть данного поведенческого паттерна заключается в установлении отображений, которые призваны обеспечить возможность для представителей Поставщика и Потребителя, а также для интеллектуальной ИТ формирования и анализа требований к ИС рассматривать представления паттерна проектирования требований к ИС на уровне данных в виде единой атрибутивной модели требований. С другой стороны, эти отображения должны минимизировать степень связности моделей отдельных структурных паттернов, чтобы снизить сложность представления паттерна проектирования требований к ИС на уровне данных в ходе выполнения различных операций интеллектуальной ИТ формирования и анализа требований к ИС.

Из всех возможных морфизмов для моделей структурных паттернов Pt_{Id}, Pt_{Ctrl} и Pt_{Op} могут существовать только морфизмы $H(Pt_{Id}, Pt_{Ctrl})$, $H(Pt_{Id}, Pt_{Op})$, $H(Pt_{Op}, Pt_{Ctrl})$ и $H(Pt_{Op}, Pt_{Id})$. Существование данных морфизмов обусловлено, как показано выше, следующими условиями:

– невозможно осуществлять управление неидентифицированным требованием к ИС и осуществлять операции над таким требованием;

– в ходе выполнения отдельных операций над требованиями к ИС возможна инициация как процесса управления требованием, так и процесса идентификации нового требования.

Морфизм $H(Pt_{Id}, Pt_{Ctrl})$ в общем случае имеет следующий вид:

$$H(Pt_{Id}, Pt_{Ctrl}) : \{ At_r^{Id}, At_{asp}^{Id}, At_{acs}^{Id} \} \to \{ At_r^{Ctrl}, At_{dsp}^{Ctrl}, At_{dcs}^{Ctrl} \}, \quad (4.30)$$

где At_r^{Id} – подмножество атрибутов, идентифицирующих требование к ИС, присутствующее в модели структурного паттерна Pt_{Id};

At_{asp}^{Id} – подмножество атрибутов, описывающих сотрудников Поставщика как возможных авторов требований к ИС, присутствующее в модели структурного паттерна Pt_{Id};

At_{acs}^{Id} – подмножество атрибутов, описывающих сотрудников Потребителя как возможных авторов требований к ИС, присутствующее в модели структурного паттерна Pt_{Id};

At_r^{Ctrl} – подмножество атрибутов, идентифицирующих требование к ИС, присутствующее в модели структурного паттерна Pt_{Ctrl};

At_{dsp}^{Ctrl} – подмножество атрибутов, идентифицирующих лиц, принимающих решение об изменении текущего статуса требования к ИС со стороны Поставщика, присутствующее в модели структурного паттерна Pt_{Ctrl};

At_{dcs}^{Ctrl} – подмножество атрибутов, идентифицирующих лиц, принимающих решение об изменении текущего статуса требования к ИС со стороны Потребителя, присутствующее в модели структурного паттерна Pt_{Ctrl}.

Морфизм $H(Pt_{Id}, Pt_{Op})$ в общем случае имеет следующий вид:

$$H(Pt_{Id}, Pt_{Op}) : \{ At_r^{Id}, At_{asp}^{Id}, At_{acs}^{Id} \} \to \{ At_r^{Op}, At_{aopsp}^{Op}, At_{aopcs}^{Op} \}, \quad (4.31)$$

где At_r^{Op} – подмножество атрибутов, идентифицирующих требование к ИС, присутствующее в модели структурного паттерна Pt_{Op};

At_{aopsp}^{Op} – подмножество атрибутов, описывающих сотрудников Поставщика, участвующих в выполнении операции над требованием к ИС, присутствующее в модели структурного паттерна Pt_{Op};

At^{Op}_{aopcs} – подмножество атрибутов, описывающих сотрудников Потребителя, участвующих в выполнении операции над требованием к ИС, присутствующее в модели структурного паттерна Pt_{Op}.

Морфизм $H(Pt_{Op}, Pt_{Id})$ в общем случае имеет следующий вид:

$$H(Pt_{Op}, Pt_{Id}) : \{ At^{Op}_{aopsp}, At^{Op}_{aopcs} \} \rightarrow \{ At^{Id}_{asp}, At^{Id}_{acs} \}. \quad (4.32)$$

Морфизм $H(Pt_{Op}, Pt_{Ctrl})$ в общем случае имеет следующий вид:

$$H(Pt_{Op}, Pt_{Ctrl}) : \{ At^{Op}_{r}, At^{Op}_{aopsp}, At^{Op}_{aopcs} \} \rightarrow \{ At^{Ctrl}_{r}, At^{Ctrl}_{dsp}, At^{Ctrl}_{dcs} \}. \quad (4.33)$$

Все морфизмы (4.29) – (4.33) являются инъективными.

Необходимо отметить, что для случая структурных паттернов проектирования требований к ИС, описывающих отдельные версии требований, морфизмы, составляющие подкласс $H(D^{Pt}_{IS})$, будут иметь аналогичный вид.

Таким образом, предлагаемые в данном подразделе модели структурных паттернов (4.8) – (4.13) как объектов подкласса D^{Pt}_{IS} и поведенческих паттернов (4.17), (4.21), (4.23) – (4.33) как морфизмов подкласса $H(D^{Pt}_{IS})$ определяет синтаксис и семантику атрибутивных моделей требований к ИС на уровне данных. С точки зрения практической реализации, разработанные модели задают структуру схемы фрагмента хранилища данных, обеспечивающего хранение представлений требований к ИС и их версий на уровне данных вне зависимости от конкретных проектов создания ИС, в которых были сформулированы эти требования.

4.4 Разработка структурных паттернов проектирования требований к информационной системе на уровне информации

Как показано в подразделе 4.2, представлением требования к ИС на уровне информации предложено считать множество возможных описаний требования в виде текстов на естественных или формальных языках, или же в виде набора целевых показателей, характеризующих степень удовлетворения требования к ИС. Однако, как показано в подразделе 1.7, подобные тексты чаще всего представляют описания требований, формулируемых в ходе инициации, планирования и исполнения проекта создания конкретной ИС. Повторное использование подобных текстов чрезвычайно затруднено.

Сказанное позволяет сделать вывод о нецелесообразности повторного использования представлений требований на уровне информации в процессах макропроектирования и, в частности, в процессе проектирования архитектуры ИС. В то же время не подвергается сомнению трактовка представлений требования к ИС как результатов выполнения работ по выявлению требований, ранее неизвестных Поставщику, Потребителю или им обоим, а также формулирования выявленных требований к ИС.

Поэтому представления требований к ИС на уровне информации следует рассматривать как набор описаний этих требований, выполненных с различных точек зрения различными участниками проекта создания ИС с целью фиксации знаний этих участников о предметной области и разрабатываемой ИС.

Данное определение понятия «представление требования к ИС на уровне информации» позволяет рассматривать любое из возможных описаний требования к ИС как частный случай публикации этого требования. ***Под термином «публикация требования к ИС» здесь и в дальнейшем следует понимать:***

а) описание условия или возможности, необходимых Потребителю ИТ-услуг для решения проблемы или достижения цели, выполненное одним из допустимых в рамках методологии разработки ИС способов;

б) описание условия или возможности, которой должна обладать ИС или компонент ИС (ИТ-услуга, ИТ-сервис) с точки зрения Поставщика или Потребителя ИТ-услуг, соответствующих договору, стандарту, спецификации или другому официальному документу, выполненное одним из допустимых в рамках методологии разработки ИС способов.

Данное определение термина «публикация требования к ИС» является частным случаем пункта «в)» определения понятия «требование к ИС», данного в подразделе 1.4. Этот частный случай применяется для описания требований к ИС, выдвигаемых в ходе выполнения текущего проекта создания ИС. Документирование требований к ИС, повторно используемых в рамках нового проекта создания ИС, может выполняться другими способами, неприменимыми для описания представлений требований к ИС на уровне информации.

Существующие методологии разработки ИС позволяют выделить следующие способы описания требований к ИС:

а) тексты на естественном языке, являющиеся результатами интервьюирования или анкетирования сотрудников Потребителя ИТ-услуг;

б) тексты на языке программирования, являющиеся результатами прототипирования компонентов ИС представителями Поставщика ИТ-услуг;

в) структурные визуальные модели, являющиеся результатами описания управляемого процесса, разрабатываемой ИС или отдельных компонентов ИС (например, модели, выполненные в нотациях IDEF0, IDEF3, DFD, ERD);

г) объектно-ориентированные визуальные модели, являющиеся результатами описания управляемого процесса или отдельных компонентов ИС (например, модели, выполненные в нотациях языка UML).

Два последних способа описания требований к ИС в настоящее время занимают промежуточное место между описаниями требований на естественном языке и описаниями требований в виде программно реализуемых прототипов.

В то же время рассмотрение представлений требования к ИС на уровне информации только как множества публикаций, выполненных одним или несколькими из перечисленных выше способов, является недостаточным для использования этих требований в проекте создания ИС и особенно для повторного использования этих требований в других проектах создания ИС. Для выделения из этих описаний знаний об управляемом объекте или процессе, а также о разрабатываемой ИС или ее компоненте необходим специальный механизм трансформации описаний требований к ИС к виду, пригодному для извлечения знаний.

Используем для разработки модели представления требования к ИС на уровне информации предложенную в подразделе 4.3 обобщенную модель сформулированных требований к ИС L_{IS}^{tr}. Тогда модель представления сформулированных требований к ИС на уровне информации следует представлять как формализованное описание каждого элемента подкласса I_{IS}^{tr} с последующим уточнением после отнесения этого элемента к одному из следующих подклассов: I_{IS}^{B}, I_{IS}^{IB}, I_{IS}^{s}, I_{IS}^{f}, I_{IS}^{nf}, I_{IS}^{fw}, I_{IS}^{nfw}. Исходя из этой точки зрения, модель представления сформулированных требований к ИС на уровне информации будет представлять собой кортеж атрибутов, который, по аналогии с моделью (4.7), также структурно разделен на две части:

$$M_{I_{IS}^{tr}} = \left\langle < M_I^{At_{Pt}} >, < M_I^{At_{gr}} > \right\rangle, \tag{4.34}$$

где $M_{I_{IS}^{tr}}$ – модель подкласса представлений сформулированных требований к ИС на уровне информации;

$< M_I^{At_{Pt}} >$ – кортеж элементов атрибутивной модели требования к ИС, которые определяются паттернами проектирования требований к ИС на уровне информации и являются обязательными для требований любой группы;

$< M_I^{At_{gr}} >$ – кортеж элементов атрибутивной модели требования к ИС, которые определяются, исходя из индивидуальных особенностей выполнения Поставщиком и Потребителем процессов, непосредственно работающих с представлениями требований конкретной группы на уровне информации.

Как уже было отмечено выше, элементы кортежа $< M_I^{At_{Pt}} >$ в общем случае определяются двумя основными структурными паттернами:

– структурным паттерном публикаций требований к ИС Pt_{r_publ};

– структурным паттерном представления публикаций требований к ИС в виде, пригодном для извлечения знаний Pt_{r_kn}.

Модель, описывающая структурный паттерн публикаций требований к ИС Pt_{r_publ}, в общем случае будет иметь следующий вид:

$$Pt_{r\_publ} =< At_r, At_{pub\_t}, At_{pub}, At_{files}, At_{proj}, < at_r, at_{pub\_t}, at_{pub}, at_{files}, at_{proj} >>, \quad (4.35)$$

где Pt_{r_publ} − модель структурного паттерна публикации требования к ИС на уровне информации, определяющего семантику описания публикаций требований к ИС различных групп;

At_{pub_t} − подмножество атрибутов, описывающих типы публикаций требований к ИС;

At_{pub} − подмножество атрибутов, описывающих публикации требований к ИС;

At_{files} − подмножество атрибутов, описывающих файлы, в которых хранятся публикации требований к ИС;

at_{pub_st} − атрибут, идентифицирующий тип публикации конкретного сформулированного требования к ИС;

at_{pub} − атрибут, идентифицирующий публикацию конкретного сформулированного требования к ИС;

at_{files} − атрибут, идентифицирующий файл, в котором хранится публикация конкретного сформулированного требования к ИС;

$< at_r, at_{pub\_t}, at_{pub}, at_{files}, at_{proj} >$ − кортеж атрибутов, описывающий факт создания публикации конкретного сформулированного требования к ИС в проекте создания конкретной ИС.

Интерпретацией модели (4.35) в ходе разработки ИТ формирования и анализа требований к ИС является схема одной или нескольких витрин данных, предназначенных для хранения исторической информации о публикациях требований, выполненных в рамках проектов создания различных ИС.

В случае использования модели (4.35) для идентификации версии требования к ИС, данная модель принимает следующий вид:

$$Pt_{r\_publ} =< At_r, At_{rv}, At_{pub\_t}, At_{pub}, At_{files}, At_{proj}, < at_r, at_{rv}, at_{pub\_t}, at_{pub}, at_{files}, at_{proj} >> . \quad (4.36)$$

Интерпретация модели (4.36) в ходе разработки ИТ формирования и анализа требований к ИС аналогична интерпретации модели (4.35).

Для описания представления публикации требования к ИС в виде, пригодном для извлечения знаний, следует, прежде всего, определить основной подход к организации этих знаний. Для этого необходимо установить возможные виды представлений требований к ИС на уровне информации.

Как показано в подразделе 2.3, формирование требований к ИС Поставщиком и Потребителем завершается созданием групп функциональных и нефункциональных требований к ИТ-услугам (на стадии макропроектирования ИС) или групп функциональных и нефункциональных требований к ИТ-сервисам (на стадии микропроектирования ИС). Поэтому общий подход к рассмотрению описаний требований и паттернов проектирования требований должен определяться необходимостью использования унифицированных представлений функциональных требований к ИТ-услугам и реализующим эти услуги ИТ-сервисам для решения задач синтеза архитектур ИС и ее обеспечивающей части.

В качестве такого общего подхода рассмотренная в разделе 3 концепция представления требований к ИС использует процессный подход. Данный подход позволяет рассматривать любое (функциональное или нефункциональное) требование к любому элементу информационной системы как часть описания процесса, визуальная модель которого в нотации Гейна-Сарсона показана на рис. 4.4.

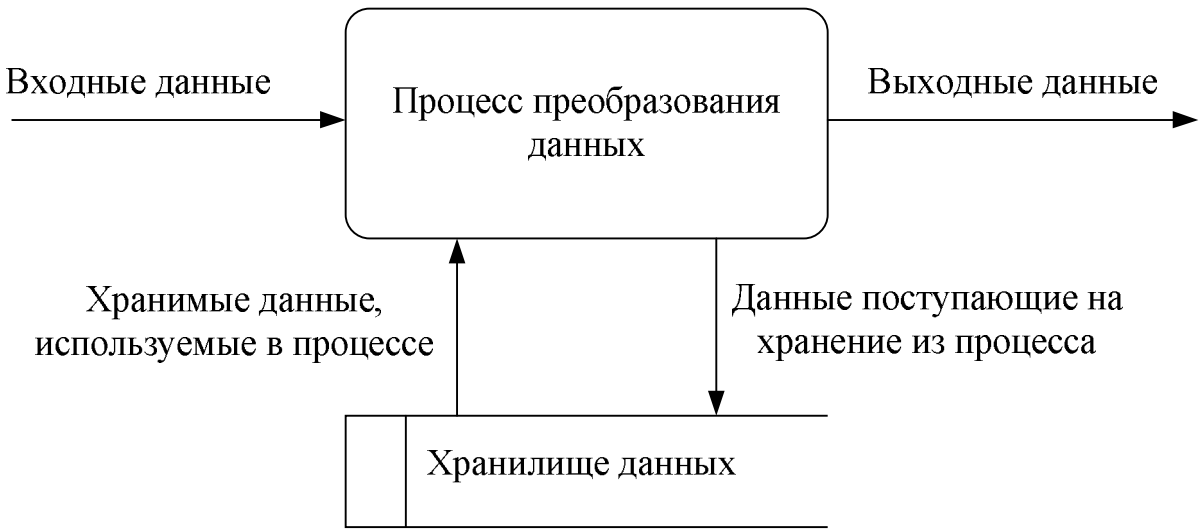

Рисунок 4.4 – Визуальная модель процесса преобразования данных как типового элемента информационной системы

Предлагаемое обобщенное представление элемента информационной системы позволяет уточнить онтологию понятия «требование к элементу информационной системы», приведенного на рис. 4.3, путем выдвижения следующих аксиоматических утверждений и следствий из них [135, 136].

Утверждение 1. Каждое требование к процессу описывает только один требующий автоматизации процесс предприятия или только одну разрабатываемую функцию ИС. По мере необходимости это описание может быть детализировано совокупностью других требований к процессам.

Утверждение 2. Требования к границам системы описывают пределы ИС, за которые нельзя выходить при выявлении требований к процессам.

Следствие из Утверждения 2. Каждое требование к процессу будет представлять собой требование к границам системы для совокупности требований к процессам, детализирующим описание этого требования.

Утверждение 3. Требования к входным, хранимым и выходным данным описывают структуры данных с устанавливаемой разработчиком степенью детализации и позволяют уточнить взаимодействие процессов, требующих автоматизации.

Следствие из Утверждения 3. Требования к входным, выходным и хранимым данным являются звеньями, согласующими описания различных требований к процессам.

Данные утверждения позволяют рассматривать понятие «требование к элементу ИС» как категориальное, включающее в себя следующие классы требований:

а) требование к входным данным, устанавливающим особенности представления потоков, инициирующих выполнение процесса;

б) требование к выходным данным, устанавливающим особенности представления результатов выполнения процесса;

в) требование к хранимым данным, устанавливающим особенности представления потоков, доступных для использования в ходе периодического выполнения процесса;

г) требование к процессу преобразования данных, который устанавливает особенности преобразований требований к входным данным в требования к хранимым данным, требований к входным данным в требования к выходным данным, а также требований к хранимым данным в требования к выходным данным в ходе выполнения процесса.

В свою очередь класс «Требование к хранимым данным» можно разделить на два следующих подкласса:

а) требование к хранимым данным, используемым в процессе;

б) требование к данным, поступающим на хранение из процесса.

Контекстная диаграмма классов, описывающая понятие «Требование к элементу информационной системы», приведена на рис. 4.5.

Предлагаемые классы, описывающие требования к элементам ИС, полностью соответствуют как главной цели деятельности ИС (см. подраздел 1.1), так и процессному подходу к описания требований к ИС.

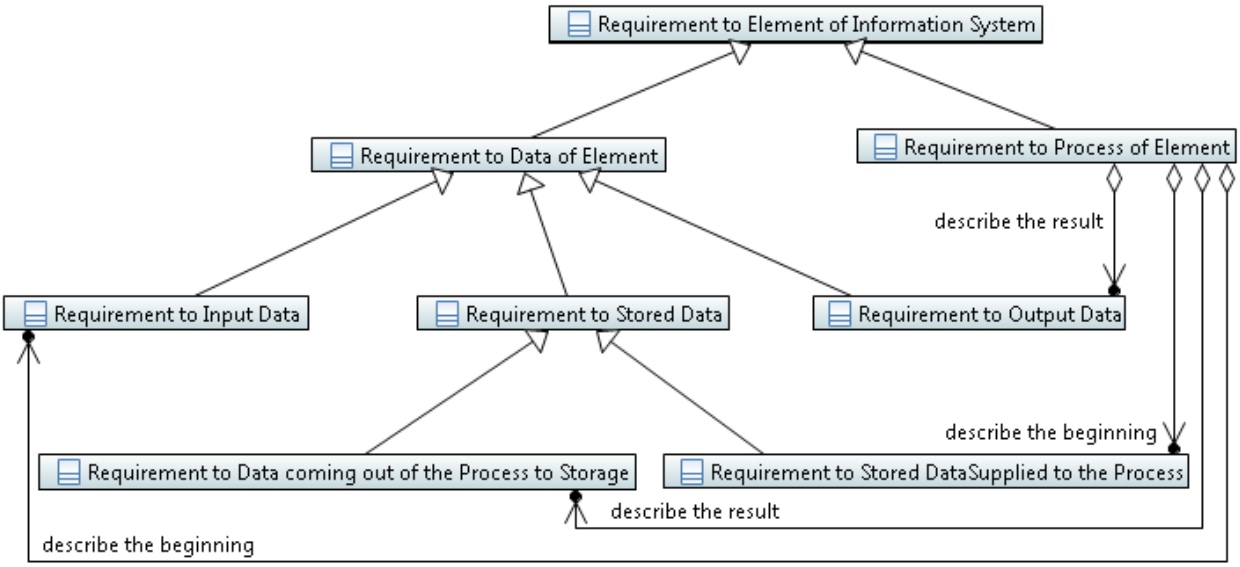

Рисунок 4.5 – Контекстная диаграмма классов, описывающая понятие «Требование к элементу информационной системы»

Рассмотренный подход к определению понятия «требование к элементу ИС» позволяет рассматривать модель структурного паттерна представления публикации требований к ИС в виде, пригодном для извлечения знаний, Pt_{r_kn} как множество, состоящее из следующих элементов:

а) модель структурного паттерна представления структур данных, присутствующих в публикации требования к ИС, $Pt_{r\_kn\_dst}$;

б) модель структурного паттерна представления внешних по отношению к процессу объектов, которые непосредственно участвуют в выполнении процесса и присутствуют в публикации требования к ИС, $Pt_{r\_kn\_eobj}$;

в) модель структурного паттерна представления процессов, присутствующих в публикации требования к ИС, $Pt_{r\_kn\_proc}$.

Модель структурного паттерна представления структур данных, присутствующих в публикации требования к ИС, $Pt_{r\_kn\_dst}$ в общем случае будет иметь следующий вид:

$$Pt_{r\_kn\_dst} =< At_{pub\_t}, At_{pub}, At_{str\_t}, At_{str}, At_{str\_m\_t}, At_{str\_m}, At_{attr\_t}, At_{attr}, < at_{pub\_t}, at_{pub}, at_{str\_t}, at_{ztr}, at_{str\_m\_t}, at_{str\_m}, at_{attr\_t}, at_{attr} >>, \quad (4.37)$$

где At_{str_t} – подмножество атрибутов, описывающих типы структур данных (сущности, классы), присутствующих в публикации требования к ИС;

At_{str} – подмножество атрибутов, описывающих структуру данных, присутствующую в публикации требования к ИС;

$At_{str\_m\_t}$ – подмножество атрибутов, описывающих типы элементов структур данных, присутствующих в публикации требования к ИС;

At_{str_m} – подмножество атрибутов, описывающих элемент структуры данных, присутствующий в публикации требования к ИС;

At_{attr_t} – подмножество атрибутов, описывающих типы атрибутов структур данных, присутствующих в публикации требования к ИС;

At_{attr} – подмножество атрибутов, описывающих атрибут структуры данных, присутствующий в публикации требования к ИС;

at_{str_t} – атрибут, идентифицирующий тип структуры данных, присутствующей в публикации конкретного сформулированного требования к ИС;

at_{str} – атрибут, идентифицирующий структуру данных, присутствующую в публикации конкретного сформулированного требования к ИС;

$at_{str\_m\_t}$ – атрибут, идентифицирующий тип элемента структуры данных, присутствующего в публикации конкретного сформулированного требования к ИС;

at_{str_m} – атрибут, идентифицирующий элемент структуры данных, присутствующий в публикации конкретного сформулированного требования к ИС;

at_{attr_t} – атрибут, идентифицирующий тип атрибута структуры данных, присутствующего в публикации конкретного сформулированного требования к ИС;

at_{attr} – атрибут, идентифицирующий атрибут структуры данных, присутствующий в публикации конкретного сформулированного требования к ИС.

Интерпретацией модели (4.37) в ходе разработки ИТ формирования и анализа требований к ИС является схема одной или нескольких витрин данных, предназначенных для хранения исторической информации о структурах данных, присутствовавших в публикациях требований, выполненных в рамках проектов создания различных ИС.

Модель структурного паттерна представления внешних по отношению к процессу объектов, которые непосредственно участвуют в выполнении процесса и присутствуют в публикации требования к ИС, $Pt_{r\_kn\_eobj}$ в общем случае будет иметь следующий вид:

$$Pt_{r\_kn\_eobj} =< At_{pub\_t}, At_{pub}, At_{eobj\_t}, At_{eobj}, < at_{pub\_t}, at_{pub}, at_{eobj\_t}, at_{eobj} >>, \quad (4.38)$$

где At_{eobj_t} – подмножество атрибутов, описывающих типы внешних по отношению к процессу объектов, которые непосредственно участвуют в выполнении процесса и присутствуют в публикации требования к ИС;

At_{eobj} – подмножество атрибутов, описывающих внешний по отношению к процессу объект, который непосредственно участвует в выполнении процесса и присутствует в публикации требования к ИС;

at_{eobj_t} – атрибут, идентифицирующий тип внешнего по отношению к процессу объекта, который непосредственно участвует в выполнении процесса и присутствует в публикации конкретного сформулированного требования к ИС;

at_{eobj} – атрибут, идентифицирующий внешний по отношению к процессу объект, который непосредственно участвует в выполнении процесса и присутствует в публикации конкретного сформулированного требования к ИС.

Интерпретацией модели (4.38) в ходе разработки ИТ формирования и анализа требований к ИС является схема одной или нескольких витрин данных, предназначенных для хранения исторической информации о внешних по отношению к процессу объектах, которые непосредственно участвуют в выполнении процесса и присутствуют в публикациях требований, выполненных в рамках проектов создания различных ИС.

Модель структурного паттерна представления процессов, присутствующих в публикации требования к ИС, $Pt_{r\_kn\_proc}$ в общем случае будет иметь следующий вид:

$$Pt_{r\_kn\_proc} =< At_{pub\_t}, At_{pub}, At_{proc}, At_{inst\_t}, At_{inst}, < at_{pub\_t}, at_{pub}, at_{proc}, at_{inst\_t}, at_{inst} >>, \quad (4.39)$$

где At_{proc} – подмножество атрибутов, описывающих процесс, присутствующих в публикации требования к ИС;

At_{inst_t} – подмножество атрибутов, описывающих типы элементов (внешних по отношению к процессу объектов или структур данных), связанных с процессом в публикации требования к ИС;

At_{eobj} – подмножество атрибутов, описывающих элементы, связанные с процессом в публикации требования к ИС;

at_{proc} – атрибут, идентифицирующий процесс, присутствующий в публикации конкретного сформулированного требования к ИС;

at_{inst_t} – атрибут, идентифицирующий тип элемента, связанного с процессом в публикации конкретного сформулированного требования к ИС;

at_{inst} – атрибут, идентифицирующий элемент, связанный с процессом в публикации конкретного сформулированного требования к ИС.

Интерпретацией модели (4.39) в ходе разработки ИТ формирования и анализа требований к ИС является схема одной или нескольких витрин данных, предназначенных для хранения исторической информации о процессах,

присутствующих в публикациях требований, выполненных в рамках проектов создания различных ИС.

Использование моделей (4.37) – (4.39) для описания представления публикации версий требований к ИС в виде, пригодном для извлечения знаний, не требует изменения вида этих моделей.

Таким образом, для случая использования в качестве элементарных описаний создаваемой системы отдельных требований к ИС подкласс объектов I_{IS}^{Pt} модели (3.39) следует рассматривать как множество структурных паттернов проектирования требований к ИС на уровне информации, устанавливающих конкретный вид элемента $< M_I^{At_{Pt}} >$ модели (4.34). Это множество паттернов в общем случае будет иметь следующий вид:

$$\begin{gathered} I_{IS}^{Pt} = \{ Pt_{r\_publ}, Pt_{r\_kn\_dst}, Pt_{r\_kn\_eobj}, Pt_{r\_kn\_proc} \} = \\ = \{ < At_r, At_{pub\_t}, At_{pub}, At_{files}, At_{proj}, < at_r, at_{pub\_t}, at_{pub}, at_{files}, \\ at_{proj} >>, < At_{pub\_t}, At_{pub}, At_{str\_t}, At_{str}, At_{str\_m\_t}, At_{str\_m}, \\ At_{attr\_t}, At_{attr}, < at_{pub\_t}, at_{pub}, at_{str\_t}, at_{ztr}, at_{str\_m\_t}, at_{str\_m}, \\ at_{attr\_t}, at_{attr} >>, < At_{pub\_t}, At_{pub}, At_{eobj\_t}, At_{eobj}, < at_{pub\_t}, at_{pub}, \\ at_{eobj\_t}, at_{eobj} >>, < At_{pub\_t}, At_{pub}, At_{proc}, At_{inst\_t}, At_{inst}, \\ < at_{pub\_t}, at_{pub}, at_{proc}, at_{inst\_t}, at_{inst} >> \}. \end{gathered} \tag{4.40}$$

Для случая использования в качестве элементарных описаний создаваемой системы отдельных версий требований к ИС подкласс объектов I_{IS}^{Pt} модели (3.39) следует рассматривать как множество структурных паттернов проектирования версий требований к ИС на уровне информации, устанавливающих конкретный вид элемента $< M_I^{At_{Pt}} >$ модели (4.34). Это множество паттернов в общем случае будет иметь следующий вид:

$$\begin{gathered} I_{IS}^{Pt} = \{ Pt_{r\_publ}, Pt_{r\_kn\_dst}, Pt_{r\_kn\_eobj}, Pt_{r\_kn\_proc} \} = \\ = \{ < At_r, At_{rv}, At_{pub\_t}, At_{pub}, At_{files}, At_{proj}, < at_r, at_{rv}, at_{pub\_t}, at_{pub}, \\ at_{files}, at_{proj} >>, < At_{pub\_t}, At_{pub}, At_{str\_t}, At_{str}, At_{str\_m\_t}, At_{str\_m}, \\ At_{attr\_t}, At_{attr}, < at_{pub\_t}, at_{pub}, at_{str\_t}, at_{ztr}, at_{str\_m\_t}, at_{str\_m}, \\ at_{attr\_t}, at_{attr} >>, < At_{pub\_t}, At_{pub}, At_{eobj\_t}, At_{eobj}, < at_{pub\_t}, at_{pub}, \\ at_{eobj\_t}, at_{eobj} >>, < At_{pub\_t}, At_{pub}, At_{proc}, At_{inst\_t}, At_{inst}, \\ < at_{pub\_t}, at_{pub}, at_{proc}, at_{inst\_t}, at_{inst} >> \}. \end{gathered} \tag{4.41}$$

Подкласс морфизмов $H(I_{IS}^{Pt})$, по аналогии с рассмотренным в подразделе 4.3 подклассом морфизмов $H(D_{IS}^{Pt})$, состоит из следующих видов морфизмов:

а) единичные морфизмы $1_{Pt_{r\_publ}}$, $1_{Pt_{r\_kn\_dst}}$, $1_{Pt_{r\_kn\_eobj}}$, $1_{Pt_{r\_kn\_proc}}$, отображающие структурные паттерны Pt_{r_publ}, $Pt_{r\_kn\_dst}$, $Pt_{r\_kn\_eobj}$ и $Pt_{r\_kn\_proc}$, соответственно, самих в себя;

б) морфизмы, устанавливающие связи между структурными паттернами Pt_{r_publ}, $Pt_{r\_kn\_dst}$, $Pt_{r\_kn\_eobj}$ и $Pt_{r\_kn\_proc}$.

Для модели паттерна Pt_{r_publ} условия существования единичных морфизмов аналогичны условиям (4.16), (4.20) и (4.22), а модели этих морфизмов будут иметь, соответственно, следующий вид:

$$1_{Pt_{r\_publ}}^{add} : [At_r \oplus At_x] \oplus [At_{pub\_t} \oplus At_x] \oplus [At_{pub} \oplus at_x] \oplus \\ \oplus [At_{files} \oplus At_x] \oplus [At_{proj} \oplus At_x] \oplus [\varnothing \oplus At_x], \tag{4.42}$$

$$1_{Pt_{r\_publ}}^{upd} : [(At_r \setminus At_x) \cup At'_x] \oplus [(At_{pub\_t} \setminus at_x) \cup At'_x] \oplus \\ \oplus [(At_{pub} \setminus At_x) \cup At'_x] \oplus [(At_{files} \setminus At_x) \cup At'_x] \oplus [(At_{proj} \setminus At_x) \cup At'_x], \tag{4.43}$$

$$1_{Pt_{r\_publ}}^{del} : [At_r \setminus At_x] \oplus [At_{pub\_t} \setminus At_x] \oplus [At_{pub} \setminus At_x] \oplus \\ \oplus [At_{files} \setminus At_x] \oplus [At_{proj} \setminus At_x]. \tag{4.44}$$

Для модели паттерна $Pt_{r\_kn\_dst}$ условия существования единичных морфизмов аналогичны условиям (4.16), (4.20) и (4.22), а модели этих морфизмов будут иметь, соответственно, следующий вид:

$$1_{Pt_{r\_kn\_dst}}^{add} : [At_{pub\_t} \oplus At_x] \oplus [At_{pub} \oplus At_x] \oplus [At_{str\_t} \oplus At_x] \oplus \\ \oplus [At_{str} \oplus At_x] \oplus [At_{str\_m\_t} \oplus At_x] \oplus [At_{str\_m} \oplus At_x] \oplus \\ \oplus [At_{attr\_t} \oplus At_x] \oplus [At_{attr} \oplus At_x] \oplus [\varnothing \oplus At_x], \tag{4.45}$$

$$1^{upd}_{Pt_{r\_kn\_dst}} : [(At_{pub\_t} \setminus At_x) \cup At'_x] \oplus [(At_{pub} \setminus at_x) \cup At'_x] \oplus \\ \oplus [(At_{str\_t} \setminus At_x) \cup At'_x] \oplus [(At_{str} \setminus At_x) \cup At'_x] \oplus \\ \oplus [(At_{str\_m\_t} \setminus At_x) \cup At'_x] \oplus [(At_{str\_m} \setminus At_x) \cup At'_x] \oplus \\ \oplus [(At_{attr\_t} \setminus At_x) \cup At'_x] \oplus [(At_{attr} \setminus At_x) \cup At'_x], \tag{4.46}$$

$$1^{del}_{Pt_{r\_kn\_dst}} : [At_{pub\_t} \setminus At_x] \oplus [At_{pub} \setminus At_x] \oplus [At_{str\_t} \setminus At_x] \oplus \\ \oplus [At_{str} \setminus At_x] \oplus [At_{str\_m\_t} \setminus At_x] \oplus [At_{str\_m} \setminus At_x] \oplus \\ \oplus [At_{attr\_t} \setminus At_x] \oplus [At_{attr} \setminus At_x]. \tag{4.47}$$

Для модели паттерна $Pt_{r\_kn\_eobj}$ условия существования единичных морфизмов аналогичны условиям (4.16), (4.20) и (4.22), а модели этих морфизмов будут иметь, соответственно, следующий вид:

$$1^{add}_{Pt_{r\_kn\_eobj}} : [At_{pub\_t} \oplus At_x] \oplus [At_{pub} \oplus At_x] \oplus [At_{eobj\_t} \oplus At_x] \oplus \\ \oplus [At_{eobj} \oplus At_x] \oplus [\varnothing \oplus At_x], \tag{4.48}$$

$$1^{upd}_{Pt_{r\_kn\_eobj}} : [(At_{pub\_t} \setminus At_x) \cup At'_x] \oplus [(At_{pub} \setminus at_x) \cup At'_x] \oplus \\ \oplus [(At_{eobj\_t} \setminus At_x) \cup At'_x] \oplus [(At_{eobj} \setminus At_x) \cup At'_x], \tag{4.49}$$

$$1^{del}_{Pt_{r\_kn\_eobj}} : [At_{pub\_t} \setminus At_x] \oplus [At_{pub} \setminus At_x] \oplus [At_{eobj\_t} \setminus At_x] \oplus \\ \oplus [At_{eobj} \setminus At_x]. \tag{4.50}$$

Для модели паттерна $Pt_{r\_kn\_proc}$ условия существования единичных морфизмов аналогичны условиям (4.16), (4.20) и (4.22), а модели этих морфизмов будут иметь, соответственно, следующий вид:

$$1^{add}_{Pt_{r\_kn\_proc}} : [At_{pub\_t} \oplus At_x] \oplus [At_{pub} \oplus At_x] \oplus [At_{proc} \oplus At_x] \\ \oplus [At_{inst\_t} \oplus At_x] \oplus [At_{inst} \oplus At_x] \oplus [\varnothing \oplus At_x], \tag{4.51}$$

$$1^{upd}_{Pt_{r\_kn\_proc}} : [(At_{pub\_t} \setminus At_x) \cup At'_x] \oplus [(At_{pub} \setminus At_x) \cup At'_x] \oplus \\ \oplus [(At_{proc} \setminus At_x) \cup At'_x] \oplus [(At_{inst\_t} \setminus At_x) \cup At'_x] \oplus \\ \oplus [(At_{inst} \setminus At_x) \cup At'_x], \tag{4.52}$$

$$1^{del}_{Pt_{r\_kn\_proc}} : [At_{pub\_t} \setminus At_x] \oplus [At_{pub} \setminus At_x] \oplus [At_{proc} \setminus At_x] \oplus \\ \oplus [At_{inst\_t} \setminus At_x] \oplus [At_{inst} \setminus At_x]. \quad (4.53)$$

Из всех возможных морфизмов, устанавливающих связи между моделями структурных паттернов Pt_{r_publ}, $Pt_{r\_kn\_dst}$, $Pt_{r\_kn\_eobj}$ и $Pt_{r\_kn\_proc}$, могут существовать только следующие морфизмы:

– $H(Pt_{r\_publ}, Pt_{r\_kn\_dst})$;

– $H(Pt_{r\_publ}, Pt_{r\_kn\_eobj})$;

– $H(Pt_{r\_publ}, Pt_{r\_kn\_proc})$;

– $H(Pt_{r\_kn\_dst}, Pt_{r\_kn\_proc})$;

– $H(Pt_{r\_kn\_eobj}, Pt_{r\_kn\_proc})$.

Существование данных морфизмов обусловлено следующими ограничениями:

– невозможно появление описаний ранее неизвестных в рамках ИТ формирования и анализа требований к ИС структур данных, внешних объектов и процессов без предварительного появления публикации требования к ИС, содержащей эти структуры данных, внешние объекты и процессы;

– невозможно подробное описание конкретного процесса без предварительного описания структур данных и внешних объектов, характерных только для данного процесса.

Отдельным вопросом, требующим дальнейшего исследования, является вопрос существования таких морфизмов:

а) $H(Pt_{r\_kn\_dst}, Pt_{r\_publ})$;

б) $H(Pt_{r\_kn\_eobj}, Pt_{r\_publ})$;

в) $H(Pt_{r\_kn\_proc}, Pt_{r\_publ})$;

г) $H(Pt_{r\_kn\_proc}, Pt_{r\_kn\_dst})$;

д) $H(Pt_{r\_kn\_proc}, Pt_{r\_kn\_eobj})$.

Существование первых трех морфизмов данного перечня может быть объяснено наличием какого-либо метода автоматического формирования публикации неизвестного Поставщику или Потребителю требования к ИС по отдельным описаниям процессов, структур данных и внешних объектов, присутствующим в публикациях требований, сформулированных или повторно используемых в ходе создания конкретной ИС. Существование четвертого и пятого морфизмов данного перечня может быть объяснено наличием какого-либо метода автоматического формирования описаний структур данных и внешних объектов на основе описания процесса, присутствующего в публикациях требований, сформулированных или повторно используемых в ходе создания конкретной ИС. Последний случай особенно интересен для решения проблемы

адаптации типовой ИС для различных объектов или процессов автоматизации, имеющих общее назначение.

Морфизм $H(Pt_{r\_publ}, Pt_{r\_kn\_dst})$ в общем случае имеет следующий вид:

$$H(Pt_{r\_publ}, Pt_{r\_kn\_dst}) : \{ At_{pub\_t}^{r\_publ}, At_{pub}^{r\_publ} \} \rightarrow \rightarrow \{ At_{pub\_t}^{r\_kn\_dst}, At_{pub}^{r\_kn\_dst} \}, \quad (4.54)$$

где $At_{pub\_t}^{r\_publ}$ – подмножество атрибутов, описывающих тип публикации требования к ИС в модели структурного паттерна Pt_{r_publ};

$At_{pub}^{r\_publ}$ – подмножество атрибутов, описывающих публикацию требования к ИС в модели структурного паттерна Pt_{r_publ};

$At_{pub\_t}^{r\_kn\_dst}$ – подмножество атрибутов, описывающих тип публикации требования к ИС в модели структурного паттерна $Pt_{r\_kn\_dst}$;

$At_{pub}^{r\_kn\_dst}$ – подмножество атрибутов, описывающих публикацию требования к ИС в модели структурного паттерна $Pt_{r\_kn\_dst}$.

Морфизм $H(Pt_{r\_publ}, Pt_{r\_kn\_eobj})$ в общем случае имеет следующий вид:

$$H(Pt_{r\_publ}, Pt_{r\_kn\_eobj}) : \{ At_{pub\_t}^{r\_publ}, At_{pub}^{r\_publ} \} \rightarrow \rightarrow \{ At_{pub\_t}^{r\_kn\_eobj}, At_{pub}^{r\_kn\_eobj} \}, \quad (4.55)$$

где $At_{pub\_t}^{r\_kn\_eobj}$ – подмножество атрибутов, описывающих тип публикации требования к ИС в модели структурного паттерна $Pt_{r\_kn\_eobj}$;

$At_{pub}^{r\_kn\_eobj}$ – подмножество атрибутов, описывающих публикацию требования к ИС в модели структурного паттерна $Pt_{r\_kn\_eobj}$.

Морфизм $H(Pt_{r\_publ}, Pt_{r\_kn\_proc})$ в общем случае имеет следующий вид:

$$H(Pt_{r\_publ}, Pt_{r\_kn\_proc}) : \{ At_{pub\_t}^{r\_publ}, At_{pub}^{r\_publ} \} \rightarrow \rightarrow \{ At_{pub\_t}^{r\_kn\_proc}, At_{pub}^{r\_kn\_proc} \}, \quad (4.56)$$

где $At_{pub\_t}^{r\_kn\_proc}$ – подмножество атрибутов, описывающих тип публикации требования к ИС в модели структурного паттерна $Pt_{r\_kn\_proc}$;

$At_{pub}^{r\_kn\_proc}$ – подмножество атрибутов, описывающих публикацию требования к ИС в модели структурного паттерна $Pt_{r\_kn\_proc}$.

Морфизм $H(Pt_{r\_kn\_dst}, Pt_{r\_kn\_proc})$ в общем случае имеет следующий вид:

$$H(Pt_{r\_kn\_dst}, Pt_{r\_kn\_proc}): \begin{cases} \{At_{str\_t}^{r\_kn\_dst}, At_{str\_m\_t}^{r\_kn\_dst}, At_{attr\_t}^{r\_kn\_dst}\} \rightarrow At_{inst\_t}^{r\_kn\_proc} \\ \{At_{str}^{r\_kn\_dst}, At_{str\_m}^{r\_kn\_dst}, At_{attr}^{r\_kn\_dst}\} \rightarrow At_{inst}^{r\_kn\_proc}, \end{cases} \quad (4.57)$$

где $At_{str\_t}^{r\_kn\_dst}$ – подмножество атрибутов, описывающих тип структуры данных в модели структурного паттерна $Pt_{r\_kn\_dst}$;

$At_{str\_m\_t}^{r\_kn\_dst}$ – подмножество атрибутов, описывающих тип элемента структуры данных в модели структурного паттерна $Pt_{r\_kn\_dst}$;

$At_{attr\_t}^{r\_kn\_dst}$ – подмножество атрибутов, описывающих тип атрибута структуры данных в модели структурного паттерна $Pt_{r\_kn\_dst}$;

$At_{inst\_t}^{r\_kn\_proc}$ – подмножество атрибутов, описывающих тип элемента, связанного с процессом, в модели структурного паттерна $Pt_{r\_kn\_proc}$;

$At_{str}^{r\_kn\_dst}$ – подмножество атрибутов, описывающих структуру данных в модели структурного паттерна $Pt_{r\_kn\_dst}$;

$At_{str\_m}^{r\_kn\_dst}$ – подмножество атрибутов, описывающих элемент структуры данных в модели структурного паттерна $Pt_{r\_kn\_dst}$;

$At_{attr}^{r\_kn\_dst}$ – подмножество атрибутов, описывающих атрибут структуры данных в модели структурного паттерна $Pt_{r\_kn\_dst}$;

$At_{inst}^{r\_kn\_proc}$ – подмножество атрибутов, описывающих элемент, связанный с процессом, в модели структурного паттерна $Pt_{r\_kn\_proc}$.

Морфизм $H(Pt_{r\_kn\_eobj}, Pt_{r\_kn\_proc})$ в общем случае имеет следующий вид:

$$H(Pt_{r\_kn\_dst}, Pt_{r\_kn\_proc}): \begin{cases} At_{eobj\_t}^{r\_kn\_eobj} \rightarrow At_{inst\_t}^{r\_kn\_proc} \\ At_{eobj}^{r\_kn\_eobj} \rightarrow At_{inst}^{r\_kn\_proc}, \end{cases} \quad (4.58)$$

где $At_{eobj\_t}^{r\_kn\_eobj}$ – подмножество атрибутов, описывающих тип внешнего по отношению к процессу объекта в модели структурного паттерна $Pt_{r\_kn\_eobj}$;

$At_{eobj}^{r\_kn\_eobj}$ – подмножество атрибутов, описывающих внешний по отношению к процессу объект в модели структурного паттерна $Pt_{r\_kn\_eobj}$.

Морфизмы (4.54) – (4.56) являются инъективными.

Необходимо отметить, что для случая структурных паттернов проектирования требований к ИС, описывающих отдельные версии требований, морфизмы, составляющие подкласс $H(I_{IS}^{Pt})$, будут иметь аналогичный вид.

Таким образом, предлагаемые в данном подразделе модели структурных паттернов (4.35) – (4.39) как объектов подкласса I_{IS}^{Pt} и поведенческих паттернов (4.42) – (4.58) как морфизмов подкласса $H(I_{IS}^{Pt})$ определяет синтаксис и семантику атрибутивных моделей требований к ИС на уровне информации. С точки зрения практической реализации, разработанные модели задают структуру схемы фрагмента хранилища данных, обеспечивающего хранение представлений требований к ИС и их версий на уровне информации вне зависимости от конкретных проектов создания ИС, в которых были сформулированы эти требования.

4.5 Разработка структурных паттернов проектирования требований к информационной системе на уровне знаний

В подразделе 4.2 представлением требования к ИС на уровне знаний предлагается считать онтологии элемента управляемого объекта или процесса, элемента ИС или ИС в целом, к которым выдвигается требование. В то же время стремление повторного использования требований к ИС обусловливает необходимость дополнительного выделения онтологий элементов ИС или ИС в целом, реализованных в выполненных ранее проектах создания ИС. В целом предлагается рассматривать следующие виды онтологий:

а) онтологии предметной области (ПрО), которые представляют собой знания об автоматизируемых объектах или БП, добытые в ходе выявления и анализа требований к ИС;

б) онтологии реализованных требований к ИС, которые представляют собой знания о структурах данных и процессах ИС, ИТ-услуг и ИТ-сервисов, созданных в рамках предыдущих проектов;

в) онтологии требований к создаваемой ИС, которые представляют собой знания о структурах данных и процессах ИС, ИТ-услуг и ИТ-сервисов, выделяемые в ходе формирования и анализа требований в рамках текущего проекта создания ИС.

Формирование этих онтологий предполагается проводить на основе фреймовой модели знаний. Применение данной модели обусловлено следующими соображениями:

– использование фреймовой модели знаний позволяет применять единый математический аппарат для описания знаний о ПрО для формального представления требований к ИС, а также для описания знаний, реализованных в элементах ИС в виде моделей ПО данной системы;

– использование фреймовой модели позволяет реализовать взаимно-однозначное отображение представлений требований к создаваемой ИС на уровне знаний в элементы ПО этой ИС.

Поскольку большинство современных СУБД основано на реляционной модели данных, применение фреймовой модели знаний позволяет осуществить в результате решения задачи объектно-реляционного отображения последующее взаимно-однозначное отображение требований к ПрО и элементов ПО в элементы ИО создаваемой ИС [137, 138].

Фрейм является структурой данных для представления стереотипной ситуации. В методологии объектно-ориентированного программирования (ООП) данное понятие соответствует классу [139].

Как правило, фреймовые модели ПрО представляются в виде сети фреймов, состоящей из узлов и различных связей между ними [139]:

$$M =< FR, C, G >, \tag{4.59}$$

где $FR = \{fr_1, ..., fr_h\}$ – множество информационных единиц (фреймов);

$C = \{C_1, C_2, ..., C_n\}$ – множество связей и отношений между информационными единицами (иерархические, ссылочные и т.д.);

G – множество отображений, которые задают связи из заданного множества $\{C_1, C_2, ..., C_n\}$ (как иерархические связи наследования, так и горизонтальные связи ассоциации фреймов) между информационными единицами, входящими во множество FR ; $G_i \in G =< fr_{i1}, fr_{i2}, C_i >$ [140].

Фреймы подразделяются на фреймы-прототипы (им соответствуют классы в ООП и таблицы в базах данных) и фреймы-экземпляры (им соответствуют экземпляры классов – объекты и записи в таблицах баз данных). По характеру отношений фреймы классифицируются следующим образом: «субфреймы, фреймы и суперфреймы – это иерархически упорядоченные элементы,

образующие системы фреймов». В технологии ООП существуют также иерархические и ссылочные связи, а понятиям «субфрейм» и «суперфрейм» соответствуют субклассы (классы-потомки) и суперклассы (классы-родители). Аналогично фреймовой модели, в виде сети можно представить иерархию классов (диаграмма классов в UML), в которой классы-родители определяют набор полей и функциональности, свойственный всем классам-потомкам [68, 139, 141-143].

Фрейм $fr \in FR$ можно описать структурированным множеством, имеющим следующий вид [141]:

$$fr = \{ n, [(ns_1, vs_1, ps_1), (ns_2, vs_2, ps_2), ..., (ns_k, vs_k, ps_k)] \}, \qquad (4.60)$$

где n – имя фрейма;

(ns, vs, ps) – слот фрейма;

k – количество слотов фрейма;

ns_i – имя слота, $i = \overline{1,k}$;

vs_i – значение слота, $i = \overline{1,k}$;

ps_i – имя присоединенной процедуры, $i = \overline{1,k}$.

В качестве значения слота «имя присоединенной процедуры» в фреймах используется подпрограмма процедурного типа. Присоединённым процедурам в ООП соответствуют методы классов, в реляционных базах данных им соответствуют связанные с таблицами триггеры, процедуры и функции [144].

Однако фреймовая модель в классическом виде, представленном выражением (4.60), не вполне соответствует особенностям представления знаний о ПрО, и особенно о создаваемой ИС и ее элементах. Подобные особенности определяются, главным образом, теми парадигмами проектирования систем, которые помещаются Поставщиком в основу подавляющего большинства проектных решений ИС в целом, отдельных ИТ-услуг и ИТ-сервисов. В качестве таких парадигм обычно указываются структурный и объектно-ориентированный подходы. Данные подходы, в свою очередь, определяют необходимость модернизации фреймовой модели представления знаний о ПрО, ИС, ИТ-услугах и ИТ-сервисах таким образом, чтобы эти знания можно было впоследствии преобразовать в паттерны и конкретные проектные решения по видам обеспечений создаваемой ИС.

Прежде всего, предлагается расширить базовое понятие «фрейм» путем выделения как отдельного элемента фрейма совокупности всех методов (присоединенных процедур) $Mt = \{mt_1, ..., mt_z\}$, связанных с фреймом в целом, а не с конкретными слотами.

Кроме того, предлагается расширить базовое понятие «фрейм» введением дополнительного понятия «интерфейс фрейма», соответствующего понятию «интерфейс класса» в методологии ООП [144]. Следует отметить, что в

некоторых языках программирования (например, в языке Object Pascal) поддерживается возможность определения свойств в интерфейсах классов. В других языках программирования (например, в языках Java, C++ и ряде других) свойства могут быть объявлены путем задания методов *GetPropertyName* и *SetPropertyName*, осуществляющих доступ к полям класса. Интерфейсы, как и классы, могут содержать ссылочные свойства и наследоваться, образовывая иерархии [68, 143, 145].

В реляционных базах данных нет понятия, полностью соответствующего термину «интерфейс класса». Наиболее близким к нему с точки зрения участия в организации взаимодействия между различными компонентами ИС является термин «представление».

Сказанное выше позволяет определить термин «интерфейс фрейма» как декларативное объявление множества свойств и методов без их подробного описания (в том числе, без подробного описания присоединенных процедур). Предполагается, что каждый интерфейс фрейма описывает отдельную точку зрения на фрейм или их подмножество, причем данная точка зрения может не воспринимать эти фрейм или подмножество фреймов полностью. Данное определение согласуется с определением понятия «архитектурная точка зрения», приведенным в подразделе 1.3.

Согласно предложенному определению, формализованное описание интерфейса фрейма if будет представлять собой структурированное множество, имеющее следующий вид [144]:

$$if =< g, \{ns\_if_1,...,ns\_if_n\}, \{nm_1,...,nm_s\} >, \quad (4.61)$$

где g – глобально уникальный идентификатор (Globally Unique IDentifier – GUID);

$\{ns\_if_1,...,ns\_if_n\}$ – множество декларативных объявлений слотов в интерфейсе фрейма;

$\{nm_1,...,nm_s\}$ – множество декларативных объявлений методов в интерфейсе фрейма.

Следует отметить, что для любого декларативно объявленного имени слота интерфейса фрейма можно поставить в соответствие множество значений как отдельного фрейма $fr \in Fr$, так и отдельного слота $ns_i \in fr$. В дальнейшем фрейм, участвующий в формировании интерфейса if, будем называть порождающим фреймом.

Процесс формирования интерфейса if из подмножества порождающих фреймов $FR_{par} \subset FR$ [14] можно описать отображением $F^{if}_{FR_{par}} : FR_{par} \to if$. Данное отображение биективно, поскольку каждый конкретный элемент

[14] Строгость выделения подмножества порождающих фреймов обусловлена нецелесообразностью появления интерфейса фрейма типа «господь бог» – то есть включающего в себя элементы всех фреймов, образующих конкретную сеть.

интерфейса if может иметь в качестве родителя только один фрейм или слот фрейма из подмножества FR_{par}.

Тогда формализованное описание понятия «фрейма» (4.60) с учетом предлагаемых модификаций примет следующий вид [144]:

$$fr = \{ n, [(ns_1, vs_1, ps_1), ..., (ns_k, vs_k, ps_k)], \{ if_1, ..., if_n \}, \{ mt_1, ..., mt_z \} \}, \quad (4.62)$$

где $\{ if_1, ..., if_n \}$ – множество интерфейсов, используемых фреймом fr (может быть пустым).

Использование модифицированного формализованного описания фрейма (4.62) позволяет утверждать, что сеть фреймов (4.59) будет включать не только знания о структурах данных, представляемые в виде фреймов, но и знания о процессах, в ходе которых будет осуществляться взаимодействие этих структур. Такие знания представляются в сети в виде интерфейсов и методов фреймов. Кроме того, использование модифицированного описания фрейма позволяет формализовать не только описания знаний о ПрО, ИО и ПО создаваемой ИС, но и описания их взаимно-однозначных отображений.

Используем для разработки модели сети фреймов как представления требования к ИС на уровне знаний предложенную в подразделе 4.3 обобщенную модель сформулированных требований к ИС L_{IS}^{tr}. Тогда модель представления требований к ИС на уровне знаний следует представлять как формализованное описание каждого элемента подкласса I_{IS}^{tr} с последующим уточнением после отнесения этого элемента к одному из следующих подклассов: I_{IS}^{B}, I_{IS}^{IB}, I_{IS}^{s}, I_{IS}^{f}, I_{IS}^{nf}, I_{IS}^{fw}, I_{IS}^{nfw}. Основываясь на этой точке зрения, модель представления сформулированных требований к ИС на уровне знаний будет представлять собой кортеж атрибутов, который, по аналогии с моделью (4.7), также структурно разделен на две части:

$$M_{K_{IS}^{tr}} = \left\langle < M_K^{At_{Pt}} >, < M_K^{At_{gr}} > \right\rangle, \quad (4.63)$$

где $M_{K_{IS}^{tr}}$ – модель подкласса представлений сформулированных требований к ИС на уровне знаний;

$< M_K^{At_{Pt}} >$ – кортеж элементов атрибутивной модели требования к ИС, которые определяются паттернами проектирования требований к ИС на уровне знаний и являются обязательными для требований любой группы;

$< M_K^{At_{gr}} >$ – кортеж элементов атрибутивной модели требования к ИС, которые определяются, исходя из индивидуальных особенностей выполнения

Поставщиком и Потребителем процессов, непосредственно работающих с представлениями требований конкретной группы на уровне знаний.

Решение о формировании рассмотренных выше видов онтологий на основе модифицированной фреймовой модели знаний (4.62) как сети фреймов (4.59) определяет необходимость выделения следующих структурных паттернов проектирования требований к ИС на уровне знаний:

а) структурный паттерн проектирования фрейма;

б) структурный паттерн проектирования интерфейса фрейма;

в) структурный паттерн проектирования связей между узлами сети фреймов;

г) обобщенный структурный паттерн проектирования сети фреймов.

Модель, описывающая структурный паттерн проектирования фрейма, Pt_{fr_str}, в общем случае будет иметь следующий вид:

$$Pt_{fr\_str} =< At_n, At_{el\_fr}, At_{el\_fr\_t}, < at_n, at_{el\_fr}, at_{el\_fr\_t} >>, \qquad (4.64)$$

где At_n – кортеж атрибутов, описывающих имя фрейма;

At_{el_fr} – кортеж атрибутов, описывающих элемент фрейма (слот, интерфейс, метод);

$At_{el\_fr\_t}$ – кортеж атрибутов, описывающих тип элемента фрейма;

at_n – атрибут, идентифицирующий имя фрейма;

at_{el_fr} – атрибут, идентифицирующий элемент фрейма;

$at_{el\_fr\_t}$ – атрибут, идентифицирующий тип элемента фрейма.

Интерпретацией модели (4.64) в ходе разработки ИТ формирования и анализа требований к ИС является схема одной или нескольких витрин данных, предназначенных для хранения исторической информации о фреймах как узлах сети фреймов, описывающей ПрО, реализованные требования к ИС или требования к создаваемой ИС.

Модель, описывающая структурный паттерн проектирования интерфейса фрейма Pt_{if}, в общем случае будет иметь следующий вид:

$$Pt_{if} =< At_g, At_{el\_if}, At_{el\_if\_t}, < at_g, at_{el\_if}, at_{el\_if\_t} >>, \qquad (4.65)$$

где At_g – кортеж атрибутов, описывающих глобально уникальный идентификатор интерфейса фрейма;

At_{el_if} – кортеж атрибутов, описывающих элемент интерфейса фрейма (слот, метод);

$At_{el\_if\_t}$ – кортеж атрибутов, описывающих тип элемента интерфейса фрейма;

at_g – атрибут, идентифицирующий глобально уникальный идентификатор интерфейса фрейма;

at_{el_if} – атрибут, идентифицирующий элемент интерфейса фрейма;

$at_{el\_fr\_t}$ – атрибут, идентифицирующий тип элемента интерфейса фрейма.

Интерпретацией модели (4.65) в ходе разработки ИТ формирования и анализа требований к ИС является схема одной или нескольких витрин данных, предназначенных для хранения исторической информации об интерфейсах фреймов как узлах сети фреймов, описывающей ПрО, реализованные требования к ИС или требования к создаваемой ИС.

Модель, описывающая структурный паттерн проектирования связей между узлами сети фреймов, Pt_{fr_rel} в общем случае будет иметь следующий вид:

$$Pt_{fr\_rel} =< At_{fr\_rel\_n}, At_{el\_fr\_rel}, At_{el\_fr\_rel\_t}, < at_{fr\_rel\_n}, at_{el\_fr\_rel}, at_{el\_fr\_rel\_t} >>, \quad (4.66)$$

где $At_{fr\_rel\_n}$ – кортеж атрибутов, описывающих имя связи;

$At_{el\_fr\_rel}$ – кортеж атрибутов, описывающих элемент описания связи;

$At_{el\_fr\_rel\_t}$ – кортеж атрибутов, описывающих тип элемента описания связи;

$at_{fr\_rel\_n}$ – атрибут, идентифицирующий имя связи;

$at_{el\_fr\_rel}$ – атрибут, идентифицирующий элемент описания связи;

$at_{el\_fr\_rel\_t}$ – атрибут, идентифицирующий тип элемента описания связи.

Описания возможных типов связей между объектными и структурными моделями сущностей приведены в [134].

Интерпретацией модели (4.66) в ходе разработки ИТ формирования и анализа требований к ИС является схема одной или нескольких витрин данных, предназначенных для хранения исторической информации о связях между узлах сетей фреймов, описывающих ПрО, реализованные требования к ИС или требования к создаваемой ИС.

Основываясь на приведенных моделях структурных паттернов проектирования элементов сети фреймов, можно сделать вывод, что модель, описывающая структурный паттерн проектирования сети фреймов (4.59), Pt_{net_fr} в общем случае будет иметь следующий вид:

$$Pt_{net\_fr} =< Pt_{fr\_str}, Pt_{if}, Pt_{fr\_rel}, < at_n^1, at_n^2, at_{if}, at_{fr\_rel\_n} >>, \quad (4.67)$$

где at_n^1 – атрибут, идентифицирующий имя первого фрейма, который может принимать участие в образовании связи (может быть не определен);

at_n^2 – атрибут, идентифицирующий имя второго фрейма, который может принимать участие в образовании связи (может быть не определен).

Атрибут at_{if} также может быть не определен, поскольку каждая конкретная связь может существовать только между двумя фреймами, или между одним фреймом и интерфейсом. Условие, ограничивающее существование связей между узлами сети фреймов, будет иметь следующий вид:

$$\begin{gathered}\forall(at_n^1, at_n^2, at_{if}) \in < at_n^1, at_n^2, at_{if}, at_{fr\_rel\_n} > \\ (at_n^1 \bigcap at_n^2) \oplus (at_n^1 \bigcap at_{if}) \oplus (at_n^2 \bigcap at_{if}) \oplus (at_{if} \bigcap at_n^1) \oplus (at_{if} \bigcap at_n^2) = 1.\end{gathered} \quad (4.68)$$

Интерпретацией модели (4.67) в ходе разработки ИТ формирования и анализа требований к ИС является схема витрины данных, предназначенной для хранения информации о сети фреймов, описывающей ПрО, реализованные требования к ИС или требования к создаваемой ИС.

Таким образом, подкласс объектов K_{IS}^{Pt} модели (3.39) следует рассматривать как множество структурных паттернов проектирования требований к ИС на уровне знаний, устанавливающих конкретный вид элемента $< M_K^{At_{Pt}} >$ модели (4.63). Это множество паттернов в общем случае будет иметь следующий вид:

$$\begin{gathered}K_{IS}^{Pt} = \{ Pt_{fr\_str}, Pt_{if}, Pt_{fr\_rel}, Pt_{net\_fr} \} = \\ = \{ < At_n, At_{el\_fr}, At_{el\_fr\_t}, < at_n, at_{el\_fr}, at_{el\_fr\_t} >>, \\ < At_g, At_{el\_if}, At_{el\_if\_t}, < at_g, at_{el\_if}, at_{el\_if\_t} >>, < At_{fr\_rel\_n}, \\ At_{el\_fr\_rel}, At_{el\_fr\_rel\_t}, < at_{fr\_rel\_n}, at_{el\_fr\_rel}, \\ at_{el\_fr\_rel\_t} >>, < at_n^1, at_n^2, at_{if}, at_{fr\_rel\_n} > \}.\end{gathered} \quad (4.69)$$

Подкласс морфизмов $H(K_{IS}^{Pt})$, по аналогии с рассмотренными в подразделах 4.3 и 4.4 подклассами морфизмов $H(D_{IS}^{Pt})$ и $H(I_{IS}^{Pt})$, состоит из следующих видов морфизмов:

а) единичные морфизмы $1_{Pt_{fr\_str}}$, $1_{Pt_{if}}$, $1_{Pt_{fr\_rel}}$, отображающие структурные паттерны Pt_{fr_str}, Pt_{if} и Pt_{fr_rel}, соответственно, самих в себя;

б) морфизмы, устанавливающие связи между структурными паттернами Pt_{fr_str}, Pt_{if}, Pt_{fr_rel} и Pt_{net_fr}.

Для структурного паттерна Pt_{net_fr} единичные морфизмы не существуют, поскольку данный паттерн наследует особенности структурных паттернов Pt_{fr_str}, Pt_{if} и Pt_{fr_rel}.

Для модели паттерна Pt_{fr_str} условия существования единичных морфизмов аналогичны условиям (4.16), (4.20) и (4.22), а модели этих морфизмов будут иметь, соответственно, следующий вид:

$$1^{add}_{Pt_{fr\_str}} : [At_n \oplus At_x] \oplus [At_{el\_fr} \oplus At_x] \oplus [At_{el\_fr\_t} \oplus At_x] \oplus \oplus [\varnothing \oplus At_x], \tag{4.70}$$

$$1^{upd}_{Pt_{fr\_str}} : [(At_n \setminus At_x) \cup At'_x] \oplus [(At_{el\_fr} \setminus At_x) \cup At'_x] \oplus \oplus [(At_{el\_fr\_t} \setminus At_x) \cup At'_x], \tag{4.71}$$

$$1^{del}_{Pt_{fr\_str}} : [At_n \setminus At_x] \oplus [At_{el\_fr} \setminus At_x] \oplus [At_{el\_fr\_t} \setminus At_x]. \tag{4.72}$$

Для модели паттерна Pt_{if} условия существования единичных морфизмов аналогичны условиям (4.16), (4.20) и (4.22), а модели этих морфизмов будут иметь, соответственно, следующий вид:

$$1^{add}_{Pt_{if}} : [At_g \oplus At_x] \oplus [At_{el\_if} \oplus At_x] \oplus [At_{el\_if\_t} \oplus At_x] \oplus \oplus [\varnothing \oplus At_x], \tag{4.73}$$

$$1^{upd}_{Pt_{if}} : [(At_g \setminus At_x) \cup At'_x] \oplus [(At_{el\_if} \setminus At_x) \cup At'_x] \oplus \oplus [(At_{el\_if\_t} \setminus At_x) \cup At'_x], \tag{4.74}$$

$$1^{del}_{Pt_{if}} : [At_g \setminus At_x] \oplus [At_{el\_if} \setminus At_x] \oplus [At_{el\_if\_t} \setminus At_x]. \tag{4.75}$$

Для модели паттерна Pt_{fr_rel} условия существования единичных морфизмов аналогичны условиям (4.16), (4.20) и (4.22), а модели этих морфизмов будут иметь, соответственно, следующий вид:

$$1^{add}_{Pt_{fr\_rel}} : [\,At_{fr\_rel\_n} \oplus At_x\,] \oplus [\,At_{el\_fr\_rel} \oplus At_x\,] \oplus \\ \oplus [\,At_{el\_fr\_rel\_t} \oplus At_x\,] \oplus [\,\varnothing \oplus At_x\,], \tag{4.76}$$

$$1^{upd}_{Pt_{fr\_rel}} : [(At_{fr\_rel\_n} \setminus At_x) \cup At'_x\,] \oplus [(At_{el\_fr\_rel} \setminus At_x) \cup At'_x\,] \oplus \\ \oplus [(At_{el\_fr\_rel\_t} \setminus At_x) \cup At'_x\,], \tag{4.77}$$

$$1^{del}_{Pt_{fr\_rel}} : [\,At_{fr\_rel} \setminus At_x\,] \oplus [\,At_{el\_fr\_rel} \setminus At_x\,] \oplus [\,At_{el\_fr\_rel\_t} \setminus At_x\,]. \tag{4.78}$$

Из всех морфизмов, устанавливающих связи между моделями структурных паттернов Pt_{fr_str}, Pt_{if}, Pt_{fr_rel} и Pt_{net_fr}, могут существовать только следующие морфизмы:

– $H(Pt_{fr\_str}, Pt_{if})$;
– $H(Pt_{fr\_str}, Pt_{fr\_rel})$;
– $H(Pt_{fr\_str}, Pt_{net\_fr})$;
– $H(Pt_{if}, Pt_{fr\_str})$;
– $H(Pt_{if}, Pt_{fr\_rel})$;
– $H(Pt_{if}, Pt_{net\_fr})$;
– $H(Pt_{fr\_rel}, Pt_{net\_fr})$.

Существование данных морфизмов обусловлено невозможностью появления описаний связей между узлами сети фреймов без предварительного появления описаний отдельных фреймов и интерфейсов, являющихся узлами этой сети.

Отдельным вопросом, требующим дальнейшего исследования, является вопрос существования морфизмов $H(Pt_{net\_fr}, Pt_{fr\_str})$, $H(Pt_{net\_fr}, Pt_{if})$ и $H(Pt_{net\_fr}, Pt_{fr\_rel})$. Существование данных морфизмов может быть объяснено наличием каких-либо методов автоматической добычи знаний на основе результатов анализа сформированной сети фреймов, описывающей требования к создаваемой ИС. Кроме того, не менее важным вопросом является вопрос существования морфизмов типа $H(Pt_{net\_fr}, Pt'_{net\_fr})$ и $H(Pt'_{net\_fr}, Pt_{net\_fr})$, где Pt_{net_fr} – структурный паттерн проектирования сети фреймов, описывающей сформулированные требования к создаваемой ИС, а Pt'_{net_fr} – структурный паттерн проектирования сети фреймов, описывающей ПрО или реализованные требования к ИС. Существование данных морфизмов может быть объяснено наличием методов автоматического

формирования или модификации сети фреймов. Данные методы могут использоваться для решения следующих задач:

а) формирование или модификация сети фреймов, описывающей сформулированные требования к создаваемой ИС, по результатам решения задач анализа соответствия конкретных сформулированных требований к создаваемой ИС реализованным требованиям к ранее разработанным ИС;

б) формирование или модификация сетей фреймов, описывающих ПрО или реализованные требования к ИС по результатам успешного завершения проекта создания ИС, знания о сформулированных и реализованных требованиях к которой описаны в виде соответствующей сети фреймов.

Решение задачи первого типа позволяет автоматизировать формирование сети фреймов, описывающей сформулированные требования к создаваемой ИС, с учетом решений о целесообразности повторного использования реализованных требований к ранее разработанным ИС и ИТ-услуг или ИТ-сервисов, реализующих эти требования. Решение задачи второго типа позволяет автоматизировать процессы расширения или модификации сетей фреймов, описывающих ПрО и реализованные требования к ИС, с учетом знаний, добытых из требований реализованного проекта создания ИС.

Морфизм $H(Pt_{fr\_str}, Pt_{if})$ в общем случае имеет следующий вид:

$$H(Pt_{fr\_str}, Pt_{if}) : \{At_n^{fr\_str}, At_{el\_fr}^{fr\_str}, At_{el\_fr\_t}^{fr\_str}\} \rightarrow \rightarrow \{At_{el\_if}^{if}, At_{el\_if\_t}^{if}\} \quad (4.79)$$

и существует только при выполнении условия

$$\forall\{At_n^{fr\_str}, At_{el\_fr}^{fr\_str}, At_{el\_fr\_t}^{fr\_str}\} [\{At_n^{fr\_str}\} \rightarrow \{At_{el\_if}^{if}\}] \oplus [\{At_{el\_fr}^{fr\_str}, At_{el\_fr\_t}^{fr\_str}\} \rightarrow \rightarrow \{At_{el\_if}^{if}, At_{el\_if\_t}^{if}\}] = 1, \quad (4.80)$$

где $At_n^{fr\_str}$ – подмножество атрибутов, описывающих имя фрейма в модели структурного паттерна Pt_{fr_str};

$At_{el\_fr}^{fr\_str}$ – подмножество атрибутов, описывающих элемент фрейма в модели структурного паттерна Pt_{fr_str};

$At_{el\_fr\_t}^{fr\_str}$ – подмножество атрибутов, описывающих тип элемента фрейма в модели структурного паттерна Pt_{fr_str};

$At^{if}_{el\_if}$ – подмножество атрибутов, описывающих элемент интерфейса фрейма в модели структурного паттерна Pt_{if};

$At^{if}_{el\_if\_t}$ – подмножество атрибутов, описывающих тип элемента интерфейса фрейма в модели структурного паттерна Pt_{if}.

Морфизм $H(Pt_{fr\_str}, Pt_{fr\_rel})$ в общем случае имеет следующий вид:

$$H(Pt_{fr\_str}, Pt_{fr\_rel}) : \{ At^{fr\_str}_{n}, At^{fr\_str}_{el\_fr}, At^{fr\_str}_{el\_fr\_t} \} \rightarrow \rightarrow \{ At^{fr\_rel}_{el\_fr\_rel}, At^{fr\_rel}_{el\_fr\_rel\_t} \}, \quad (4.81)$$

где $At^{fr\_rel}_{el\_fr\_rel}$ – подмножество атрибутов, описывающих элемент описания связи в модели структурного паттерна Pt_{fr_rel};

$At^{fr\_rel}_{el\_fr\_rel\_t}$ – подмножество атрибутов, описывающих тип элемента описания связи в модели структурного паттерна Pt_{fr_rel}.

Морфизм $H(Pt_{fr\_str}, Pt_{net\_fr})$ в общем случае имеет следующий вид:

$$H(Pt_{fr\_str}, Pt_{net\_fr}) : \{ At^{fr\_str}_{n}, At^{fr\_str}_{el\_fr}, At^{fr\_str}_{el\_fr\_t} \} \rightarrow \rightarrow \{ At^{net\_fr}_{n}, At^{net\_fr}_{el\_fr}, At^{net\_fr}_{el\_fr\_t} \}, \quad (4.82)$$

где $At^{net\_fr}_{n}$ – подмножество атрибутов, описывающих имя фрейма в модели структурного паттерна Pt_{net_fr};

$At^{net\_fr}_{el\_fr}$ – подмножество атрибутов, описывающих элемент фрейма в модели структурного паттерна Pt_{net_fr};

$At^{net\_fr}_{el\_fr\_t}$ – подмножество атрибутов, описывающих тип элемента фрейма в модели структурного паттерна Pt_{net_fr}.

Морфизм $H(Pt_{if}, Pt_{fr\_str})$ в общем случае имеет следующий вид:

$$H(Pt_{if}, Pt_{fr\_str}) : \{ At^{if}_{g} \} \rightarrow \{ At^{fr\_str}_{el\_fr} \}, \quad (4.83)$$

где At_g^{if} – подмножество атрибутов, описывающих глобально уникальный идентификатор интерфейса фрейма в модели структурного паттерна Pt_{if}.

Морфизм $H(Pt_{if}, Pt_{fr\_rel})$ в общем случае имеет следующий вид:

$$H(Pt_{if}, Pt_{fr\_rel}) : \{ At_g^{if}, At_{el\_if}^{if}, At_{el\_if\_t}^{if} \} \rightarrow \\ \rightarrow \{ At_{el\_fr\_rel}^{fr\_rel}, At_{el\_fr\_rel\_t}^{fr\_rel} \}. \quad (4.84)$$

Морфизм $H(Pt_{if}, Pt_{net\_fr})$ в общем случае имеет следующий вид:

$$H(Pt_{if}, Pt_{net\_fr}) : \{ At_g^{if}, At_{el\_if}^{if}, At_{el\_if\_t}^{if} \} \rightarrow \\ \rightarrow \{ At_g^{net\_fr}, At_{el\_if}^{net\_fr}, At_{el\_if\_t}^{net\_fr} \}, \quad (4.85)$$

где $At_g^{net\_fr}$ – подмножество атрибутов, описывающих глобально уникальный идентификатор интерфейса фрейма в модели структурного паттерна Pt_{net_fr};

$At_{el\_if}^{net\_fr}$ – подмножество атрибутов, описывающих элемент интерфейса фрейма в модели структурного паттерна Pt_{net_fr};

$At_{el\_if\_t}^{net\_fr}$ – подмножество атрибутов, описывающих тип элемента интерфейса фрейма в модели структурного паттерна Pt_{net_fr}.

Морфизм $H(Pt_{fr\_rel}, Pt_{net\_fr})$ в общем случае имеет следующий вид:

$$H(Pt_{fr\_rel}, Pt_{net\_fr}) : \{ At_{fr\_rel\_n}^{fr\_rel}, At_{el\_fr\_rel}^{fr\_rel}, At_{el\_fr\_rel\_t}^{fr\_rel} \} \rightarrow \\ \rightarrow \{ At_{fr\_rel\_n}^{net\_fr}, At_{el\_fr\_rel}^{net\_fr}, At_{el\_fr\_rel\_t}^{net\_fr} \}, \quad (4.86)$$

где $At_{fr\_rel\_n}^{net\_fr}$ – подмножество атрибутов, описывающих имя связи в модели структурного паттерна Pt_{net_fr};

$At_{el\_fr\_rel}^{net\_fr}$ – подмножество атрибутов, описывающих элемент описания связи в модели структурного паттерна Pt_{net_fr};

$At_{el\_fr\_rel\_t}^{net\_fr}$ – подмножество атрибутов, описывающих тип элемента описания связи в модели структурного паттерна Pt_{net_fr}.

Необходимо отметить, что для случая использования отдельных версий требований к ИС морфизмы, составляющие подкласс $H(K_{IS}^{Pt})$, будут иметь аналогичный вид.

Таким образом, предлагаемые в данном подразделе модели структурных паттернов (4.64) – (4.67) как объектов подкласса K_{IS}^{Pt} и поведенческих паттернов (4.70) – (4.86) как морфизмов подкласса $H(K_{IS}^{Pt})$ определяют синтаксис и семантику описаний знаний, добытых из публикаций требований к ИС. С точки зрения практической реализации разработанные модели задают структуру схемы фрагмента хранилища данных, обеспечивающего хранение представлений требований к ИС и их версий на уровне знаний вне зависимости от конкретных проектов создания ИС, в которых были сформулированы эти требования.

4.6 Выводы

Основываясь на изложенной в разделе 3 концепции представления требований к ИС, в данном разделе авторы рассматривают концептуальное представление архитектурных фреймворков создания ИС. Поскольку полностью формализовать описания процессов создания ИС и правил их выполнения в настоящее время не представляется возможным, предлагается выделить в рамках архитектурного фреймворка формальную часть, которая в ходе практической реализации данного фреймворка будет описывать банк моделей, методов, знаний, правил и алгоритмов, доступных для использования любой из возможных ИТ формирования и анализа требований к ИС. Такой подход позволил формализованно описать данную часть архитектурного фреймворка выражением (4.1)

На основе предложенного описания формальной части архитектурного фреймворка создания ИС было предложено определение понятия «архитектурный фреймворк быстрой разработки информационной системы» и определено место АФУР ИС в процессах, непосредственно работающих с требованиями, а также в процессе проектирования архитектуры системы. Предложена модель АФУР ИС, в которой набор видов моделей, используемых фреймворком, представляет собой совокупность структурных паттернов проектирования требований к ИС.

В соответствии с определением понятия «архитектурный фреймворк», приведенным в подразделе 1.3, разработанная модель АФУР ИС обладает следующими особенностями:

а) данная модель установлена для процесса определения требований правообладателей, процесса анализа требований и процесса проектирования

архитектуры системы в соответствии с положениями стандарта ISO/IEC 15288:2002;

б) данная модель установлена для всех представителей Поставщика и Потребителя, участвующих в процессах, указанных в п. а);

в) данная модель однозначно идентифицирует факт принятия Поставщиком решения об использовании сервисного подхода в процессах создания ИС;

г) данная модель ограничена проблемой повторного использования требований к ИС при проектировании ИС различного назначения, а также проблемой синтеза архитектуры конкретной ИС на уровне ИТ-услуг в ходе создания данной ИС;

д) данная модель основывает архитектурные точки зрения на ИС и ее элементы на концепции представления требований к ИС, сформулированной в разд. 3, которая устанавливает изначальное многообразие представлений требований к ИС в виде данных, информации и знаний;

е) данная модель содержит элементы, устанавливающие правила соответствий, интегрирующие упоминаемые в п. д) архитектурные точки зрения на ИС и ее элементы;

ж) для данной модели определены условия ее применимости в процессах, указанных в п. а);

и) данная модель согласована с положениями концептуальной модели стандарта ISO/IEC/IEEE 42010.

В целом разработанная модель устанавливает общие особенности и ограничения практик создания, интерпретации, анализа и использования AD ИС на уровне требований, выдвигаемых к данной ИС, в процессах предпроектного обследования и синтеза единого целостного AD ИС в виде ее ФС.

Анализ диаграмм классов, описывающих ядро понятия «описание архитектуры» и архитектурный фреймворк (рис. 1.2 и 1.5) показывает, что основным фактором, определяющим все последующие особенности разрабатываемой модели сервисного архитектурного фреймворка, является принятие решения об уровне представления ИС в рамках этого фреймворка. Результатом такого решения может быть одна из следующих альтернатив:

- реализация в рамках фреймворка общесистемного уровня представления ИС;
- реализация в рамках фреймворка представления ИС на уровне ИТ-услуг;
- реализация в рамках фреймворка представления ИС на уровне ИТ-сервисов.

В данном исследовании было принято решение о реализации в рамках фреймворка представления ИС на уровне ИТ-услуг. Данное решение обусловлено описанными в [50] особенностями процессов, для которых разрабатывается модель.

Для успешной реализации структурных паттернов как основополагающего компонента модели формальной части АФУР ИС были определены основные способы представления требований к ИС на уровнях данных, информации и знаний. Исходя из предложенных способов, были сформулированы основные способы представления

паттернов проектирования требований к ИС на соответствующих уровнях. В качестве основной точки зрения на данные паттерны предлагается представлять их метамоделями, определяющими синтаксис и семантику представлений требований различных групп на уровнях данных, информации и знаний.

Для разработки формализованного описания структурных паттернов проектирования требований к ИС была предложена обобщенная модель сформулированного требования к ИС в виде категории (4.3). На основе данной модели были разработаны следующие модели структурных и поведенческих паттернов проектирования требований к ИС:

а) модели структурных паттернов (4.8) – (4.13) как объектов подкласса D_{IS}^{Pt} и поведенческих паттернов (4.17), (4.21), (4.23) – (4.33) как морфизмов подкласса $H(D_{IS}^{Pt})$, определяющие синтаксис и семантику атрибутивных моделей требований к ИС на уровне данных;

б) модели структурных паттернов (4.35) – (4.39) как объектов подкласса I_{IS}^{Pt} и поведенческих паттернов (4.42) – (4.58) как морфизмов подкласса $H(I_{IS}^{Pt})$, определяющие синтаксис и семантику атрибутивных моделей требований к ИС на уровне информации;

в) модели структурных паттернов (4.64) – (4.67) как объектов подкласса K_{IS}^{Pt} и поведенческих паттернов (4.70) – (4.86) как морфизмов подкласса $H(K_{IS}^{Pt})$, определяющие синтаксис и семантику описаний знаний, добытых из публикаций требований к ИС.

В основу предложенных моделей была положена многомерная модель данных, позволяющая наилучшим способом хранить и обрабатывать данные и метаданные, характеризующие ПрО, ранее разработанные ИС и создаваемую ИС. В то же время предложенные модели могут быть трансформированы в классические схемы реляционных баз данных для упрощения практической реализации интеллектуальной ИТ формирования и анализа требований к ИС.

Следует отметить, что представление моделей структурных паттернов проектирования требований к ИС в виде кортежей позволяет рассматривать эти паттерны как формализованные высказывания определенных требований к способам организации описаний требований к ИС как объектов, автоматизированно обрабатываемых в рамках интеллектуальной ИТ формирования и анализа требований к ИС. Такое представление позволяет использовать подобную ИТ для постепенной адаптации структурных и поведенческих паттернов проектирования требований к ИС к особенностям конкретной организации Поставщика. Однако проблема ограничений целостности системы структурных и поведенческих паттернов, допускающих существование такой системы, ее модификацию и расширение, в данной работе не рассматривается и требует дополнительных теоретических исследований.

5 ИНТЕЛЛЕКТУАЛЬНАЯ ИНФОРМАЦИОННАЯ ТЕХНОЛОГИЯ УСКОРЕННОЙ РАЗРАБОТКИ ИНФОРМАЦИОННЫХ СИСТЕМ

5.1 Интеллектуальная информационная технология ускоренной разработки информационных систем: основные определения

Приведенное в конце подраздела 1.1 определение понятия «информационная технология» является результатом уточнения определения, изложенного в стандарте ГОСТ 34.003-90 [1]. Однако данный стандарт не учитывает особенности процессного представления жизненного цикла ИС и сервисного подхода к созданию ИС. Поэтому прежде, чем перейти к описанию особенностей практической реализации интеллектуальной ИТ формирования и анализа требований к ИС, возможное подмножество которых описано моделью (3.44), следует вначале уточнить определение понятия «информационная технология».

Первое из этих уточнений связано с ПрО, для которой формулируется понятие. В общем случае, как показано в подразделе 1.1, ИТ используется для выполнения одной или нескольких функций ИС или, как следует из [1], функций сбора, хранения, обработки, передачи и использования данных. Однако, говоря о процессах создания ИС, можно говорить только о существовании системы управления эти процессами. Утверждение о существовании единой целостной ИС, используемой для автоматизации управления этими процессами, является весьма спорным и нуждается в доказательстве для каждого конкретного проекта создания ИС. Поэтому говорить о каких-либо функциях в определении множества возможных ИТ создания ИС нельзя.

Второе уточнение связано с общностью точек зрения на архитектурный фреймворк и ИТ как на совокупность методов, приемов и способов. Разница этих точек зрения заключается в следующем аспекте:

а) архитектурный фреймворк, как показано в подразделах 1.3 и 4.1, представляет собой совокупность методов, приемов и способов, используемых для выполнения создания ИС в целом на основе конкретного подхода;

б) ИТ представляет собой совокупность унифицированных методов, приемов и способов применения комплекса средств автоматизации или его отдельных элементов, которая применяется для формальной обработки данных в ходе выполнения работ одного или нескольких процессов создания ИС в соответствии с принятым Поставщиком и Потребителем архитектурным фреймворком.

Иными словами, ИТ следует рассматривать как формальную часть архитектурного фреймворка, описываемую моделью (4.1). При этом каждый

архитектурный фреймворк может быть реализован множеством различных ИТ, не противоречащих базовым положениям этого фреймворка.

Исходя из сказанного выше, любую ИТ, используемую в ходе создания ИС, следует рассматривать как совокупность унифицированных методов, приемов и способов применения комплекса средств автоматизации или его отдельных элементов, используемую для выполнения одной или нескольких работ в рамках процессов создания ИС.

Термин «Интеллектуальная ИТ формирования и анализа требований к ИС» был введен в подразделе 3.4 для описания любой возможной ИТ, в основе которой находится модель (3.44). Исходя из сказанного выше, данный термин следует трактовать как совокупность унифицированных и основанных на знаниях методов, приемов и способов применения комплекса средств автоматизации или его отдельных элементов, используемую для выполнения одной или нескольких работ в рамках процессов создания ИС, непосредственно работающих с требованиями к данной системе.

Тогда для реализации одной из интеллектуальных ИТ формирования и анализа требований к ИС на основе разработанных в разделе 4 моделей следует в общем случае разработать ***совокупность унифицированных в рамках АФУР ИС и основанных на знаниях методов, приемов и способов применения комплекса средств автоматизации или его отдельных элементов, используемую для выполнения одной или нескольких работ в рамках процессов создания ИС, непосредственно работающих с требованиями к данной системе.*** Такую совокупность методов, приемов и способов здесь и в дальнейшем будем называть ***интеллектуальной ИТ ускоренной разработки ИС***.

5.2 Основные ИТ-услуги разрабатываемой информационной технологии

Как показано в подразделе 5.1, цель, назначение и состав ИТ-услуг, предлагаемых любой ИТ создания ИС, определяется совокупностью процессов создания ИС, декларированных в [50]. Применительно к интеллектуальной ИТ ускоренной разработки ИС, ее цель, назначение и состав ИТ-услуг будут определяться следующими процессами создания ИС:

а) процесс определения требований правообладателей;

б) процесс анализа требований;

в) процесс проектирования архитектуры.

В ходе выполнения работ данных процессов цели Потребителя и Поставщика не всегда совпадают, а в отдельных аспектах могут быть противоположными. Поэтому задачи оптимизации как архитектуры ИС, так и самих процессов создания ИС часто решаются индивидуально, на основе опыта и интуиции Поставщика, с учетом специфики конкретного проекта создания

ИС, архитектуры, функциональной структуры и предполагаемого набора хранимых данных ИС. Структуры данных, хранимых и обрабатываемых в ИС, платформы, средства разработки, применяемые паттерны проектирования системы и её программного обеспечения (архитектурные стили), а также непосредственно архитектура системы определяются требованиями предприятия-заказчика и спецификой конкретного БП, элементом которого станет создаваемая ИС.

Одним из наиболее распространённых способов решения указанных выше задач путем сокращения трудозатрат на разработку ИС и других программных систем является повторное использование компонентов, полученных в результате выполнения предыдущих проектов. Предпосылками использования этого способа являются:

- высокая трудоемкость и сложность разработки современных ИС;
- высокая стоимость выполнения проектных работ;
- длительные сроки выполнения проекта ИС при разработке «с нуля»;
- наличие у большинства организаций-Поставщиков портфелей ранее выполненных проектов создания ИС и других программных систем.

Основным недостатком этого способа является сложность анализа портфеля ранее выполненных проектов создания ИС на предмет возможности применения компонентов этих ИС при разработке новой системы. Кроме того, задачи максимизации объёма повторного использования готовых компонентов в ходе создания новой ИС и обеспечения максимального соответствия создаваемой ИС специфике конкретного БП объекта автоматизации во многом являются противоречивыми. Для решения данных задач необходим анализ ПрО и требований к проектируемой ИС, а также их сопоставления с терминами различных ПрО и требованиями к разработанным ранее ИС, что позволит определить возможность полного или частичного применения имеющихся готовых решений в создаваемой ИС. От решения данных задач, осуществляемого, как правило группами экспертов, зависит эффективность повторного использования готовых компонентов и, как следствие, степень снижения затрат ресурсов на выполнение проекта, а также эффективность проектируемой ИС в целом. Решение данных задач требует хорошего знания аналитиками как ПрО, для которой разрабатывается ИС, так и ПрО, для которых были выполнены предыдущие проекты. Очевидно, что в этом случае значительно возрастает роль указанных выше процессов создания ИС и уровень качества результатов их выполнения.

Поэтому как с теоретической, так и практической точек зрения актуальной является задача разработки интеллектуальной ИТ быстрого создания ИС, обеспечивающей автоматизацию проведения анализа ПрО и портфеля выполненных проектов на предмет возможности применения полученных ранее решений в ходе создания новой ИС, что позволит повысить эффективность повторного использования компонентов ИС за счет снижения сложности

сопоставления терминов различных ПрО и устранения ошибок при проведении подобного сопоставления.

Для решения данной задачи в рамках интеллектуальной ИТ ускоренной разработки ИС предлагается реализовать следующие ИТ-услуги:

а) выявление требований к ИС;

б) формирование иерархий терминов ПрО создаваемой ИС;

в) сравнение сформированных иерархий терминов ПрО ИС с разработанными ранее иерархиями терминов ПрО;

г) выделение и уточнение паттернов требований к ИС;

д) синтез архитектуры ИС на основе выделенных паттернов требований к ИС;

е) осуществление взаимно-однозначного отображения паттернов ИС в сущности базы данных и классы программного обеспечения разрабатываемой ИС.

Ниже рассмотрим суть указанных ИТ-услуг.

ИТ-услуга «Выявление требований к ИС» представляет собой совокупность методов, приемов и способов сбора требований к создаваемой ИС вне зависимости от способа представления этих требований, а также их последующего анализа, обработки и формализованного описания. Основным результатом данной ИТ-услуги следует считать сформированные представления требований к создаваемой ИС на уровне информации и на уровне данных, необходимые для последующего их отображения в представления требований на уровне знаний и создания онтологии анализируемой ПрО.

ИТ-услуга «Формирование иерархий терминов ПрО создаваемой ИС» представляет собой совокупность методов, приемов и способов непосредственного создания на основе различных описаний требований к ИС онтологии терминов ПрО создаваемой ИС. В общем случае онтологии терминов ПрО, выделенных из требований к создаваемой ИС, описываемые выражениями (4.59) и (4.62), представляют собой иерархии структур данных, правил их обработки и интерфейсов взаимодействия (спецификаций наборов входных и выходных данных). При этом онтология одной ПрО, соответствующей БП предприятия, как правило, имеет несколько корневых узлов, то есть характеризуется несколькими иерархиями понятий, совместно используемыми для достижения целей ИС и имеющими между собой горизонтальные связи. Это утверждение наглядно демонстрируют иерархии бизнес-классов ПО ИС, в которых отдельные иерархии наследуемых классов, как правило, связаны отношениями ассоциации (агрегации или композиции) и характеризуются наличием ситуаций их совместного использования.

Аналогичным образом предлагается описывать и онтологии каждого элемента библиотеки готовых компонентов ИС (паттернов реализованных требований). При этом онтологии ПрО создаваемой ИС и онтологии элементов реализованных ИС имеют принципиальные отличия. В онтологии ПрО создаваемой ИС отражаются термины и требования, характерные для конкретного объекта автоматизации и его БП. Эта онтология может быть избыточна, допускает

синонимию и неоднозначность описаний отдельных терминов и структур и не предполагает обязательное отражение в ИС в неизменном виде. В то же время онтология элементов реализованной ИС является системоцентричным описанием реализованных паттернов и описывает структуры важных для ИС данных (хранимых и обрабатываемых), а также правила их обработки. В данном описании избыточность и неоднозначность не допускаются, поскольку любое его изменение требует модификации элементов разработанных решений по информационному и программному обеспечениям ИС.

На основе концепции представления требований к ИС (раздел 3) представление онтологий создаваемой ИС, отражающих определённую ПрО, можно описать как универсум понятий и терминов, используемых в данной ИС. Универсум реализованных требований к ИС, описывающий библиотеку реализованных элементов ИС, является универсумом понятий, терминов, структур данных, доступных разработчику при разработке новых ИС на базе существующих систем. Над этими универсумами возможно выполнение операций расширения (введения в онтологию новых терминов) и сужения (исключения неиспользуемых терминов). Критерием добавления (или удаления) в онтологию иерархии терминов ПрО является наличие (отсутствие) горизонтальных понятийных связей (аналог отношения ассоциации в диаграмме классов) между анализируемым термином и остальными элементами онтологии.

Таким образом, применение теории фреймов для описания иерархий онтологий ПрО и реализованных элементов ИС позволяет создать их формальное описание в виде знаний, обеспечить их легкую интерпретацию Потребителем и Поставщиком в виде визуальных диаграмм классов и определить формальное условие необходимости присутствия каждого отдельного элемента в онтологии требований к ИС. Данное условие применимо как в ходе формирования онтологии создаваемой ИС на основе онтологий реализованных ранее требований к ИС, так и в ходе расширения библиотеки реализованных требований к ИС по результатам выполнения очередного проекта разработки ИС.

Основным результатом данной ИТ-услуги следует считать множество сформированных представлений требований к создаваемой ИС на уровне знаний, выделение из этих представлений универсума требований к проектируемой ИС и универсума библиотеки реализованных требований, описанных с применением фреймового представления знаний. Выполнение данной ИТ-услуги завершает подготовку исходных данных для сопоставления различных ПрО.

ИТ-услуга «Сравнение сформированных иерархий терминов ПрО ИС с разработанными ранее иерархиями терминов ПрО» представляет собой совокупность методов, приемов и способов анализа и сопоставления сформированных иерархий терминов ПрО проектируемой ИС с разработанными ранее иерархиями терминов ПрО. Целями такого анализа и сопоставления является поиск ответов на следующие вопросы:

а) какие существующие элементы ИС и в каком объеме могут быть применены;

б) в каком объеме необходима разработка новых или расширение (дополнение) существующих элементов ИС.

Выполнение данной ИТ-услуги позволяет сформулировать и включить в план проекта создания ИС основные проектные работы, осуществить последующую структурную декомпозицию работ, формировать графики выполнения работ, назначить исполнителей и т.п., то есть позволяет автоматизировать процессы планирования проекта создания ИС.

Основная сложность анализа онтологий различных ПрО связана с проблемой синонимии понятий, когда одно и то же, по сути, явление реального мира, характеризующееся схожим набором атрибутов, в разных ПрО (или даже в пределах одной ПрО) имеет несколько различных названий. Набор атрибутов и их названий в различных ПрО также может отличаться (синонимия атрибутов). В то же время, приведение названий атрибутов и объектов различных ПрО к единому глоссарию позволяет осуществлять операции пересечения множеств атрибутов, описывающих различные понятия ПрО, и оценивать степень их подобия.

Таким образом, основными результатами данной ИТ-услуги следует считать следующие группы решений, принимаемых Поставщиком и Потребителем:

а) решения о наследовании терминов (добавление терминов к существующим деревьям онтологий в роли ветвей или листьев) и, соответственно, о повторном использовании существующих элементов ИС;

б) решения о формировании новых узлов иерархий онтологий и, соответственно, о разработке новых элементов.

ИТ-услуга «Выделение и уточнение паттернов требований к ИС» представляет собой совокупность методов, приемов и способов выполнения предварительных работ, необходимых для синтеза варианта конфигурации создаваемой ИС. Полностью формализовать процесс принятия решений о наследовании терминов невозможно. Оценка подобия структур данных позволяет только сформировать положительную или отрицательную рекомендацию, а собственно принятие решения о применении термина, как и при анализе синонимов терминов остается за человеком-экспертом. Таким образом, основным результатом данной ИТ-услуги следует считать доработку существующих или создание новых аналитических описаний паттернов требований в виде онтологий, представленных иерархиями фреймов.

ИТ-услуга «Синтез архитектуры ИС на основе выделенных паттернов требований к ИС» представляет собой совокупность методов, приемов и способов синтеза слабосвязной сервис-ориентированной архитектуры ИС с применением существующих, доработанных или новых описаний паттернов требований к ИС, а также реализаций этих паттернов в виде отдельных ИТ-услуг и соответствующих ИТ-сервисов. Применение паттернов требований, каждый из

которых описывается конечной онтологией взаимосвязанных терминов ПрО, в значительной степени влияет на процесс проектирования архитектуры ИС. Каждый паттерн требования предполагает свою автономную реализацию применительно, в первую очередь, к корневым терминам деревьев онтологий, имеющих наибольший уровень абстракции. Конечной целью формирования паттернов требования является не максимальное соответствие ПрО создаваемой ИС (для учета специфики ПрО используются листья и ветви в деревьях онтологий), а максимально полное раскрытие базового узлового понятия онтологии. Следовательно, проектирование паттернов требований, реализуемых в виде набора автономных ИТ-услуг и соответствующих ИТ-сервисов, наиболее целесообразно и даёт максимальный эффект с точки зрения сервисного подхода к созданию ИС и, в частности, с точки зрения АФУР ИС.

Таким образом, основным результатом данной ИТ-услуги следует считать синтезированные варианты AD создаваемой ИС, которые позволяют описывать систему одновременно и как единый целостный объект исследования, и как совокупность ИТ-услуг, предлагаемых в рамках создаваемой ИС, а также как совокупность ИТ-сервисов, реализующих заявленные ИТ-услуги.

ИТ-услуга «Осуществление взаимно-однозначного отображения паттернов ИС в сущности базы данных и классы программного обеспечения разрабатываемой ИС» представляет собой совокупность методов, приемов и способов выполнения работ по физическому проектированию ИС в соответствии с сформированным ранее планом проекта создания данной ИС. В рамках данной ИТ-услуги осуществляется взаимно-однозначное отображение паттернов требований к ИС в сущности базы данных и классы программного обеспечения создаваемой ИС для уточняемых или новых паттернов требований, а также организация их взаимодействия в составе конфигурации создаваемой ИС.

Основными результатами данной ИТ-услуги следует считать описания схемы базы данных и классов прикладного программного обеспечения ИС, сформированные на основе выделенных ранее паттернов требований к ИС, в том числе – по результатам повторного использования элементов библиотеки готовых компонентов ИС.

Из приведенного описания ИТ-услуг интеллектуальной ИТ ускоренной разработки ИС следует необходимость совместного применения представлений требований к ИС на уровне информации, на уровне данных и на уровне знаний. Для реализации такого совместного применения в рамках данной ИТ предлагается разработать базу файлов с описаниями требований на уровне информации, базу данных и базу знаний о требованиях к ИС. Тогда схема взаимодействия представлений требований к ИС на уровнях информации, данных и знаний будет иметь вид, показанный на рис. 5.1.

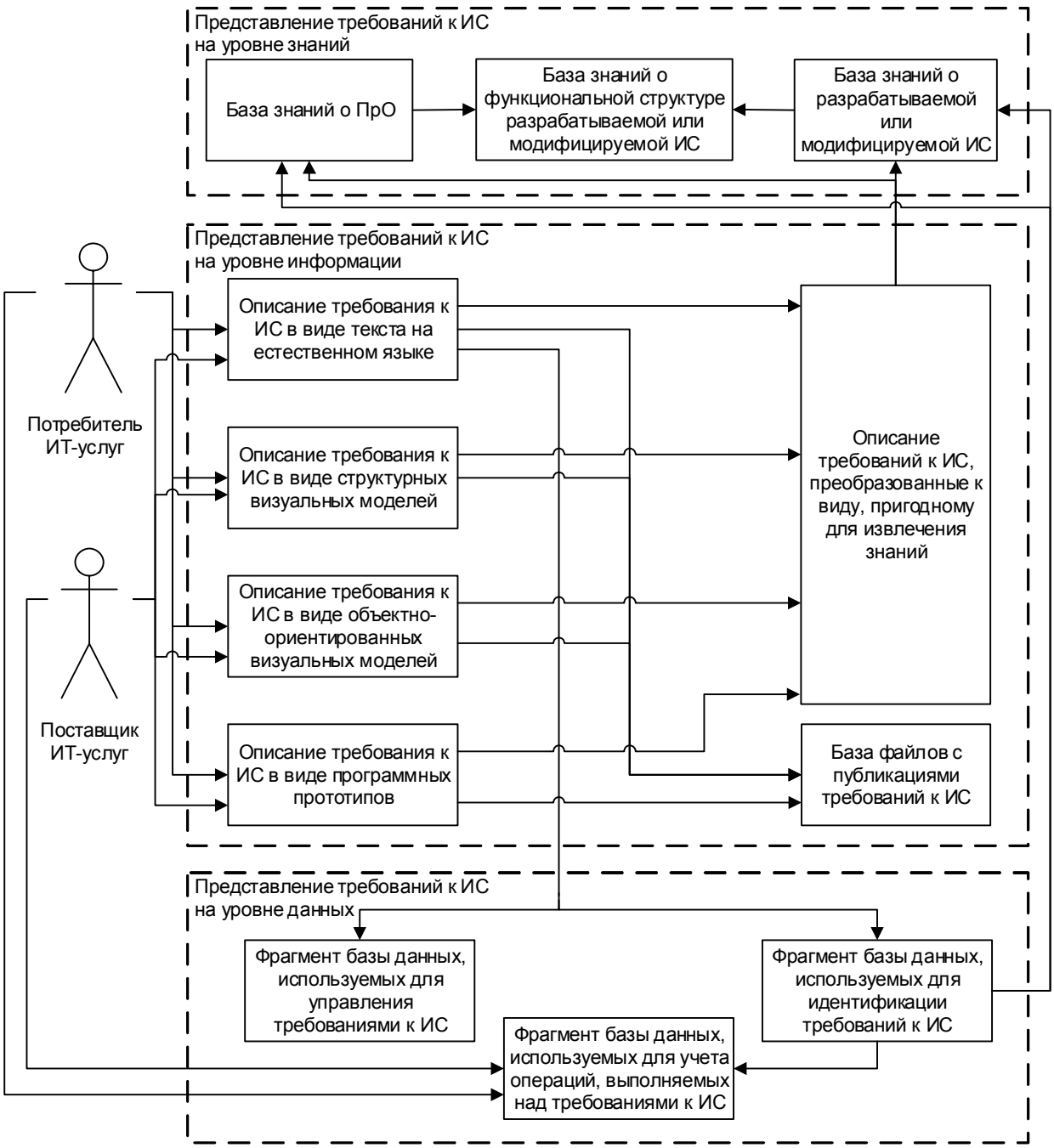

Рисунок 5.1 – Схема взаимодействия представлений требований к ИС на уровнях информации, данных и знаний в рамках интеллектуальной информационной технологии ускоренной разработки информационных систем

На основе результатов рассмотрения указанных выше ИТ-услуг интеллектуальной ИТ ускоренной разработки ИС можно сделать вывод, что отрицательными сторонами применения данной ИТ являются:

а) повышение сложности выполнения процесса анализа требований к ИС;

б) повышение необходимого уровня качества ведения проектной документации на создаваемую ИС (в частности, возникает необходимость

формирования и поддержания в актуальном состоянии описаний библиотек паттернов требований).

Положительными сторонами разрабатываемой ИТ следует считать:

а) формализацию определения возможности повторного использования элементов библиотеки существующих компонентов ИС в ходе создания новой ИС;

б) автоматизацию работ по расширению библиотеки существующих компонентов ИС;

в) снижение сложности решения задач обновления версий ИС и сопровождения вариантов тиражируемых ИС, внедренных в БП различных предприятий.

5.3 Обработка представлений требований на уровне информации

Основным представлением требований, которым оперирует как Поставщик, так и Потребитель, является представление требований на уровне информации. Данное представление может включать как неформализованные текстовые и графические описания, так и формализованные описания, созданные с применением различных языков визуального моделирования (таких как UML, нотации IDEF, ARIS и т.п.). Обработка такой слабоструктурированной информации позволяет извлечь данные, необходимые для формирования сети фреймов требований к ИС и её последующего анализа.

Анализ и извлечение данных и знаний из текстовых описаний требований является слабоформализуемой задачей и в данном исследовании рассматриваться не будет. Основой для формирования представлений требований на уровне информации следует считать предположение, согласно которому любое текстовое описание требований может быть достоверно и полно представлено в виде набора основных визуальных диаграмм, понятных одновременно и Поставщику, и Потребителю (все виды UML-диаграмм, а также основные виды структурных диаграмм: IDEF0, DFD, IDEFX3, ERD и т.п.). Аналогичным способом могут быть описаны требования, уже реализованные в виде элементов ИС и ее видов обеспечений.

Данное предположение позволяет рассматривать задачу отображения представлений требований к ИС на уровне информации в представления требований к ИС на уровне знаний как задачу обработки диаграмм визуальных моделей с целью извлечения из них знаний и формирования на их основе фреймовой модели требования.

Большинство современных CASE-средств, используемых для создания визуальных моделей, применяемых при проектировании ИС, позволяют экспортировать данные о визуальных моделях в документы формата XML и предоставляют формальные описания DOM таких документов в виде DTD или XSD.

Данные способы формального описания визуальных моделей позволяют автоматизировать их обработку с целью извлечения знаний.

Необходимость обработки различных видов визуальных моделей (структурных, объектных, отражающих статические понятия и динамические процессы), множество различных видов диаграмм, выполненных с использованием различных алфавитов построения визуальных моделей, требует приведения всего многообразия используемых диаграмм к единому алфавиту описания их элементов [146]. Такое преобразование алфавитов визуальных моделей уже упоминалось в подразделе 3.3 как один из способов реализации функторов, в частности – функторов модели интеллектуальной ИТ формирования и анализа требований к ИС (3.44). Данный подход позволит автоматизировать извлечение знаний из визуальных моделей требований к ИС и заполнение базы знаний ИТ.

Анализ нотаций наиболее распространенных видов визуальных моделей позволяет выделить следующий набор общих понятий, составляющий единый алфавит их описания: «экземпляр», «сущность», «сообщение» и «атрибут». Остальные элементы визуальных моделей являются частными случаями, расширяющими приведенные базовые. В таком случае фрагмент схемы данных, описывающий визуальные модели с применением единого набора базовых понятий будет иметь вид, приведенный на рис. И.1 Приложения И.

Следует отметить, что предлагаемый на рис. И.1 фрагмент является одним из вариантов реализации рассмотренных в подразделе 4.4 структурных паттернов, определяющих подход к описанию представлений требований к ИС на уровне информации. На данной схеме опущен ряд незначительных элементов, серьезно не изменяющих суть предлагаемого решения.

Упомянутая выше необходимость применения единого алфавита обусловливает формирование описания множества диаграмм визуальных моделей требований к ИС с использованием единого набора базовых элементов, то есть наполнение схемы данных, описывающей набор визуальных диаграмм, применяемых в проекте и описывающих отдельные элементы ПрО или ИС. Поскольку исходным набором данных для этого описания являются XML-документы с открытыми XSD или DTD, извлечение из них данных и наполнение схемы данных, приведенной на рис. И.1, являются инженерными задачами, предполагающими обработку DOM XML-документа и формирование на основе данных, содержащихся в XML-документе, запросов на заполнение базы данных. Данные задачи могут быть реализованы, например, средствами XSL (см. Приложение Б).

Рассмотрим схему данных, приведенную на рис. И.1, более подробно. Первая группа таблиц, выделенная в фрагмент схемы APP_PROJECT, содержит общие сведения о проекте разрабатываемой ИС. Базовым понятием ИТ-проекта (таблица Project) является ИТ-сервис (таблица Service), который рассматривается как составная часть ИТ-услуги и в свою очередь может быть декомпозирован на отдельные функциональные операции введения,

сохранения, обработки и формирования выходных данных (таблицы Functional_operations, Functional_operations_types). Таблицы данной схемы могут быть автоматически заполнены при обработке IDEF0, EDEF3, Use case и Activity диаграмм (таблица Diagramm) соответствующего уровня абстракции (таблица File_in_project).

Следует отметить отсутствие прямой взаимосвязи между таблицами Project и Service – важным аспектом предложенной ИТ является особенность подхода к созданию ИС, согласно которой первичным понятием является не проект, а сервис. Как уже отмечалось в подразделе 5.2, акценты проектных работ фокусируются не на создаваемой ИС, а на максимально качественной разработке отдельных независимых ИТ-сервисов, из которых впоследствии собирается ИС.

Поскольку каждая диаграмма публикуется как отдельный файл, описание диаграммы как файла приведено в таблице Files схемы APP_PUBLICATION. При этом более расширенное описание диаграммы как источника знаний приведено в таблицах схемы APP_DIAGRAMM.

Таблицы Diagramm и DiagrammTypes описывают виды диаграмм и CASE-средства в которых они были сформированы. Связь диаграммы с ИТ-сервисами и отдельными функциональными операциями отображают таблицы Described_tasks и Described_operations соответственно.

Содержимое визуальных диаграмм с применением единого набора базовых элементов (общего алфавита) описывается объектами схем APP_INSTANCE, APP_ENTITY, APP_ATTRIBUTE и APP_MESSAGE. Наборы заполняемых таблиц зависят от типа диаграммы.

Наиболее абстрактным базовым понятием является «экземпляр» (таблица Instance), которое описывает объекты, используемые как в статических, та и поведенческих диаграммах. Наличие «экземпляра» на диаграмме и его роль в ней отображают таблицы Instance_in_diagramm, External_entity (является слабой сущностью, производной от таблицы Instance при эмуляции наследования в реляционной базе данных) и Role_kinds.

Понятие «сущность» наследуется от понятия «экземпляр» и расширяет его. В схеме APP_ENTITY под сущностями понимаются записи в таблицах БД и экземпляры классов ПО (таблица Classes, которая является слабой сущностью, производной от таблицы Instance), которые включают наборы операций и атрибутов (таблицы Classes_members и Classes_attributes). В свою очередь, операции сущностей делятся на процедуры и функции (таблица Member_types) и также могут иметь наборы атрибутов (таблица Classes_member_attributes), которые описываются и классифицируются на входные, выходные и хранимые в фрагменте схемы APP_ATTRIBUTE (таблицы Attribute и Attribute_type).

Таблица Class_types используется для классификации классов и сущностей (сильные и слабые сущности, абстрактные классы, интерфейсы и т.д.).

На диаграммах, описывающих бизнес-процессы или поведение системы, взаимодействие «экземпляров» между собой осуществляется с помощью

«сообщений» (фрагмент схемы APP_MESSAGE). Подвидом сообщений (таблица Message) является сообщение между классами, предполагающее синхронный или асинхронный (таблица Message_kind) вызов определенного метода класса (таблица Operation, которая является слабой сущностью, производной от таблицы Message). Такое сообщение может привести к изменению состояния объекта класса и значений его свойств (таблица Changed_members).

Рассмотренная схема данных является частью схемы данных интеллектуальной ИТ ускоренной разработки ИС и обеспечивает унификацию набора структурных элементов основных видов визуальных диаграмм, автоматизацию их обработки и наполнения схемы данными, полученными в результате анализа диаграмм. Извлечение данных из диаграмм обеспечивает возможность автоматического формирования SQL-запросов на сохранение метаданных о требованиях в процессе анализа представлений требований на уровне информации, но не исключает и участие аналитика. Соответственно, автоматизация обработки диаграмм значительно повышает требования к качеству ведения проектной документации при разработке ИС, её актуальности и достоверности.

5.4 Обработка представлений требований на уровне данных

В рамках интеллектуальной ИТ ускоренной разработки ИС представление требований на уровне данных можно разделить на два подвида. Первый подвид – это аналитические требования, которые формируются в результате обработки представлений требований к создаваемой ИС на уровне информации. Второй подвид – это представления ранее реализованных требований, доведенных до практической реализации в виде элементов ИС. При этом каждый подвид требований на уровне данных описывается набором записей в таблицах базы данных, отражающих определённые характеристики требований (по аналогии с атрибутивными моделями требований, рассмотренными в подразделе 1.5): названия требований, входные, выходные и хранимые атрибуты, описания процессов обработки данных (в соответствии с рис. 4.4). Фрагмент схемы данных, реализующей подобное описание, приведен на рис. И.2 Приложения И.

Для удобства восприятия на рис. И.2 был использован следующий подход к расположению таблиц схемы данных:

а) таблицы, описывающие аналитические требования, расположены над разделяющей чертой;

б) таблицы, описывающие ранее реализованные требования, расположены под разделяющей чертой;

в) черта пересекает таблицы, используемые при описании двух подвидов требований одновременно.

Следует отметить, что предлагаемый на рис. И.2 фрагмент является одним из вариантов реализации рассмотренных в подразделе 4.3 структурных паттернов, определяющих подход к описанию представлений требований к ИС на уровне данных. На данной схеме опущен ряд незначительных элементов, серьезно не изменяющих суть предлагаемого решения.

При отображении требований, представленных на уровне информации, в аналитические требования, представленные на уровне данных, осуществляется формирование иерархий аналитических описаний требований. При этом введенное ранее понятие «экземпляр» отображается в понятие «аналитическое требование», которое в свою очередь является производным от понятия «публикация». Другими словами, при создании представления требования на уровне данных оно публикуется в системе управления требованиями (таблица Analitical_requirement , которая является слабой сущностью, производной от таблицы Publication). При этом таблицы Publication и Analitical_requirement (рис. И.2) заполняются на основе таблицы Instance и её производных. Как и любая публикация, аналитическое требование имеет состояние, определяющее набор допустимых операций, которые с ним могут осуществляться (таблица Publication_status).

Аналитические требования, в свою очередь, подразделяются на функциональные и нефункциональные (таблица Requirement_group). Из представления требования на уровне информации также повторно используются сведения о входных и выходных атрибутах требований (таблицы Attribute и Attribute_type), которые также характеризуются типами данных (таблица Data_type). Любое требование к ИС определяет особенности механизмов и алгоритмов обработки входных и формирования выходных данных. Сведения об этих особенностях, как правило, описываются визуальными диаграммами только на уровне представления требований в виде информации. Их отображение в представление требований на уровне данных нецелесообразно.

Алгоритмы, описанные соответствующими визуальными диаграммами, реализуются непосредственно в коде программного и информационного обеспечений ИС. Поэтому представление требований на уровне данных расширяется набором файлов с упомянутыми диаграммами, прикреплёнными к публикации требования (таблицы Files и Publication_files). Таблицы приведенного на рис. И.2 фрагмента схемы APP_PUBLICATION используются для реализаций функций систем управления содержимым (Content management System – CMS) в составе интеллектуальной ИТ ускоренной разработки ИС.

Файлы с визуальными моделями также являются «публикациями» и одновременно выступают в роли источников аналитического требования (таблица Requirement_source, которая является слабой сущностью, производной от таблицы Publication; таблица Requirement_sources, обеспечивающая связь с множеством версий требований множества источников, на основе которых они были созданы). Источники требований могут быть опубликованы любым

пользователем ИТ (таблица Users, является слабой сущностью, производной от таблицы Person) независимо от того, является ли он сотрудником Поставщика или нет. Таблицы приведенного на рис. И.2 фрагмента схемы APP_USER применяются для разграничения прав доступа к публикуемым объектам ИТ.

Кроме того, аналитические требования могут дополняться и уточняться, образуя иерархические и горизонтальные связи между требованиями (таблицы Analitical_relationship и Relation_type).

Поскольку одно и то же аналитическое требование может использоваться и уточняться в нескольких проектах, на рис. И.2 отражена связь аналитического требования с проектами (таблицы Project и Requirement_in_project). Эти сведения также могут быть получены из представления требований на уровне информации. Таблицы фрагмента схемы данных APP_PROJECT применяются для описания проекта и его структурной декомпозиции при проектировании системы.

Для описания особенностей реализации управления требованиями к ИС с помощью представлений требований на уровне данных необходимо определить понятие жизненного цикла требования, этапы которого будут отражены в системе.

Создание ИС, независимо от выбранной модели жизненного цикла, методологии проектирования или архитектурного фреймворка, включает процессы формирования и анализа требований к системе. Инициация разработки новой системы означает старт нового проекта (таблица Project). В ходе анализа объекта автоматизации и описаний требований к проектируемой системе на уровне информации формируются аналитические требования (Analitical_requirement). Аналитические требования формулируются, на первых этапах анализа ПрО, поэтому дальнейшей стадией жизненного цикла аналитического требования является уточнение требования и формирование различных версий аналитических требований (Analitical_requirement_version). При этом работа с этими версиями требований требует реализации в ИТ некоторых функций, характерных для систем контроля версий. Кроме того, при повторном использовании одних и тех же требований в различных проектах также могут создаваться различные версии требования, в результате чего аналитическое требование приобретает черты, характерные для нескольких проектов. Для описания аналитических требований применяются таблицы фрагмента схемы APP_REQUIREMENT_AN, а для описания версий требований – таблицы фрагмента схемы APP_REQUIREMENT_VERS.

Наиболее поздняя и актуальная версия аналитического требования является его текущим описанием. Как и в системе контроля версий, у каждой версии требования есть разработчики (таблица Requirement_version_developers), которые могут являться как сотрудниками Поставщика (таблица Person_physical, которая является слабой сущностью, производной от таблицы Person), так и сотрудниками Потребителя (таблицы Outer_person, которая является слабой сущностью, производной от таблицы Person_physical, и Person_juridical, которая является

слабой сущностью, производной от таблицы Person). Таблицы фрагмента схемы APP_PERSON используются для описания сведений о физических и юридических лицах, участвующих в процессе формирования и разработки требования.

Поскольку каждое требование характеризуется набором входных, хранимых и выходных атрибутов, набор и характеристики которых могут изменяться от версии к версии, в интеллектуальной ИТ ускоренной разработки ИС атрибуты (таблица Attribute) также имеют версии (Analitical_attribute_version) и характеризуют не требование в целом, а версию аналитического требования (Attribute_in_requirement).

Одно и то же требование в терминах ПрО различных ИС может иметь различные названия. Поэтому в состав ИТ также входит функция управления синонимами названий требований и их атрибутов (таблицы Requirement_synonym и Attribute_synonym). Для описания атрибутов требований применяются объекты фрагмента схемы APP_ATTRIBUTE.

Как и в системе управления версиями, в рассматриваемой ИТ необходима реализация аудита действий пользователей, осуществляемых при формировании и уточнении аналитических требований (таблицы Operation registry и Requirement_operation_type).

Следующей стадией жизненного цикла требования является его реализация в качестве элементов обеспечений ИТ-сервиса ИС, то есть, непосредственная разработка элементов системы. На этой стадии аналитическому требованию ставится в соответствие его практическая реализация в виде прототипа – ИТ-сервиса ИС или его элемента.

Поскольку интеллектуальная ИТ ускоренной разработки ИС ориентирована на создание, в первую очередь, сервис-ориентированных ИС со слабыми внутрисистемными связями отдельных компонентов системы, базовым понятием такой системы является ИТ-сервис (таблица Service, которая является слабой сущностью, производной от таблицы Instance). Как отдельные требования, так и ИТ-сервисы или даже ИТ-услуги в целом могут быть использованы повторно в различных проектах. При этом принимается утверждение о том, что одно и то же требование может быть реализовано одним и только одним ИТ-сервисом системы. Это утверждение отражено в таблице Requirement (рис. И.2), которая описывает реализованное требование. Эта таблица также является слабой сущностью, производной от таблицы Instance. Перечень атрибутов, созданных при разработке требований, отражен в таблице Implemented_attribute, которая является слабой сущностью по отношению к таблице Analitical_attribute_version. Связь реализованных атрибутов и требований с аналитическими (таблица Analitical_sources) отражает механизм физической реализации требований на базе одной или нескольких аналитических версий, состояния которых по окончанию разработки фиксируются.

Аналогично аналитическим требованиям в ИТ учитываются связи между реализованными требованиями (таблица Relationship) и разработчиками

физически реализованных требований (таблица Requirement_developers). Описание реализованных требований отражается в таблицах фрагмента схемы APP_REQUREMENT.

Реализованные требования тоже не являются статичными. При разработке новых проектов реализованные требования также могут модифицироваться и совершенствоваться. Таблицы фрагмента схемы APP_REQUREMENT содержат актуальное состояние библиотеки реализованных требований, то есть описание универсума требований Поставщика. При внедрении ИС (таблица Implementation), реализующей заданный Потребителем набор требований (таблица Requirement_in_implementation) с заданным набором атрибутов (Attribute_in_implementation) фиксируется внедряемое подмножество универсума требований Поставщика, актуальное на момент внедрения. Это позволяет автоматизировать расширение и обновление внедренных версий ИС при развитии типовой системы Поставщика, то есть обеспечить повторное использование требований при развитии и сопровождении внедренной системы. Для описания требований, реализованных в версиях ИС, внедрённых на конкретных объектах автоматизации, используются таблицы фрагмента схемы APP_IMPLEMENTATION.

Объекты фрагмента схемы APP_INTEGRITY и их аналоги в фрагменте схемы APP_IMPLEMENTATION применяются для описания ограничений целостности данных, реализуемых с помощью триггеров, и используются при объектно-реляционном отображении представлений требований на уровне знаний в элементы программного и информационного обеспечений ИС, что описано в подразделе 5.6.

Последующие стадии ЖЦ реализованного требования предполагают его применение и развитие в ходе создания новых ИС в качестве элемента библиотеки готовых компонентов Поставщика.

Таким образом, рассмотренные в подразделе 1.6 состояния требований к ИС, определяемые жизненным циклом требования, должны быть расширены. Для предлагаемой интеллектуальной ИТ ускоренной разработки ИС жизненный цикл требования к ИС, как следует из приведенного выше, включает следующие стадии:

- стадия инициаций проекта ИС организацией-разработчиком;
- стадия описания требований на уровне информации в виде текстовых документов, визуальных диаграмм и т.п.;
- стадия отображения требований в представления на уровне данных и знаний, а также формирования первых версий аналитических требований;
- стадия анализа и уточнения аналитических требований, их атрибутов и связей между ними (в том числе, формирования новых версий аналитических требований);
- стадия разработки реализуемых требований на базе фиксированных версий аналитических требований;

- стадия внедрения реализованных требований и фиксации описаний внедрённых требований;
- стадия формирования библиотеки реализованных требований для их повторного использования;
- стадия повторного использования и развития аналитических и реализованных требований в ходе создания новых ИС, развития и сопровождения внедренных систем;
- стадия удаления нереализованных или неиспользуемых аналитических и реализованных требований или их версий.

Данные стадии определяют необходимость включения в интеллектуальную ИТ ускоренной разработки ИС некоторые функции систем управления проектами, систем управления содержимым и систем контроля версий (или предполагает интеграцию с системами данных классов), а именно:

- управление сведениями о проектах разработки ИС и их структурной декомпозиции (функциональность систем управления проектами);
- управление текстовыми и графическими описаниями на уровне информации в виде публикуемых в системе представителями Поставщика и Потребителя файлов (функциональность CMS-систем);
- учет и контроль версий аналитических и внедрённых требований и их атрибутов, а также осуществляемых над ними операций (функциональность систем контроля версий).

Компоненты ИТ, реализующие эти функции, также основаны на предложенных на рис. И.1 и И.2 фрагментах схемы данных.

5.5 Обработка представлений требований на уровне знаний

В подразделе 4.5 было предложено применение сетей фреймов в качестве основного описания представлений требований к ИС на уровне знаний, которое, при необходимости, может дополняться описанием требований в виде моделей отношений, семантических сетей, каузальных сценариев и т.д.

Формирование представлений требований в виде знаний возможно как для аналитических, так и для реализованных требований. На основе таблиц фрагментов схемы APP_REQUIREMENT_AN, APP_REQUIREMENT_VERS и APP_ATTRIBUTE (для аналитических требований) и таблиц фрагментов схемы APP_REQUIREMENT, APP_ATTRIBUTE (для реализованных требований), содержащих сведения о требованиях, их атрибутах и связях между ними, осуществляется формирование сети фреймов, описывающих требования в виде знаний.

Как уже отмечалось выше, требование в ходе создания сервис-ориентированных ИС реализуется в виде отдельного ИТ-сервиса, обеспечивающего автоматизацию работы с данными об определённом множестве

объектов ПрО. Поэтому одному требованию физически соответствуют набор классов ПО, таблиц базы данных и сеть фреймов, описывающих требование в виде знаний. Следовательно, физическая реализация требования требует его декомпозиции на отдельные подтребования, физически представляемые в виде отдельных классов, таблиц и т.п.

Таким образом, в интеллектуальной ИТ ускоренной разработки ИС при формировании представления требований в виде знаний осуществляются такие преобразования данных:

а) содержимое таблиц Aanalitical_requirement_version и Requirement отображается в описания названий фреймов и иерархических связей между ними;

б) содержимое таблиц Analitical_relationship и Relationship отображается в описания горизонтальных связей между фреймами;

в) содержимое таблиц Attribute, Analitical_attribute_version, Attribute_in_requirement и Implemented_attribute отображается в описания атрибутов фреймов.

Сведения об особенностях механизмов и алгоритмов обработки данных, регламентируемых требованием к ИС, могут быть извлечены из визуальных структурных и объектных моделей, описывающих требования на уровне информации (рис. И.1). Их описание в виде знаний может быть реализовано с применением каузальных сценариев. В таком случае информационные единицы сети фреймов, описывающей требование, могут быть дополнены сигнатурами присоединенных процедур фреймов, описанными в виде сценариев (см. подразделы 5.7, 5.8).

Наличие иерархических связей между объектами ПрО обусловливает образование фреймами деревьев, которые дополнительно могут быть соединены между собой горизонтальными связями. Полученная таким образом сеть фреймов, описывающая требования, будет иметь вид «низкорослого леса» или «кустарника», в соответствии со связями объектов ПрО, для которой проектируется ИС.

Во время анализа ПрО и формирования требований к ИС осуществляется декомпозиция этих требований на подтребования и выделение отдельных объектов ПрО, представлениями которых должна оперировать ИС. Реализованные требования перед выполнением работ по созданию элементов ИС, как правило, группируются вокруг одного корневого понятия ПрО и могут включать деревья понятий, детализирующих базовое дерево и связанных с ним горизонтальными связями. При этом требование может считаться новым, а не уточняющим существующее, если в сети фреймов оно является корневым узлом нового дерева объектов ПрО. Такое дерево детализируется во время дальнейшего анализа ПрО и формирования требований, уточняющих это дерево. В программном обеспечении это отражается иерархией классов-наследников одного базового (часто абстрактного) класса, в информационном обеспечении – схемами данных типа «звезда» или «снежинка», в которых

консольные таблицы и таблицы измерений сгруппированы вокруг базовой таблицы фактов, которую они расширяют и уточняют. Для удобства работы с такими наборами таблиц они могут объединяться в одну схему данных (принадлежащую одному пользователю). Примерами таких схем данных на рис. И.1, И.2 являются фрагменты APP_PERSON, APP_PROJECT и т.п., названия которых идентичны наименованиям соответствующих таблиц фактов.

Характерным свойством подобной схемы данных является возможность денормализации и объединения всех таблиц схемы в одно универсальное отношение, описывающее базовое понятие ПрО, соответствующее таблице фактов витрины данных. Например, такое универсальное отношение может быть создано в виде представления (виртуальной таблицы). Применение универсальных отношений позволяет изменять масштаб моделирования требований к ИС, оперируя реализованными требованиями как на уровне подтребований, отдельных таблиц базы данных и классов ПО, так и на уровне отдельных требований к ИС, физически представленными ИТ-сервисами или их компонентами.

Здесь следует отметить, что предлагаемая интеллектуальная ИТ ускоренной разработки ИС является в определённой степени самоописательной. Она была применена при разработке описанных выше решений, необходимых для реализации данной технологии.

Рассмотрим структуры фрагментов схемы базы данных ИТ, представленных на рис. И.1 и И.2, реализующих описанные выше требования. Как можно увидеть, приведенные на этих рисунках схемы разделены на отдельные фрагменты (множества объектов базы данных, принадлежащие одному пользователю), соответствующие реализованным требованиям к ИТ, а именно: APP_PROJECT, APP_DIAGRAMM, APP_PUBLICATION, APP_INSTANCE, APP_ENTITY, APP_ATTRIBUTE, APP_MESSAGE, APP_REQUIREMENT_AN, APP_REQUIREMENT_VERS, APP_REQUIREMENT, APP_IMPLEMENTATION, APP_INTEGRITY, APP_PERSON, APP_USER. Каждый из этих фрагментов схем организован по типу витрины данных, в которой обязательно присутствует таблица фактов, и могут присутствовать уточняющие или расширяющие её таблицы. Аналогичным образом будут выглядеть и диаграммы классов ПО ИТ.

Такая группировка классов и таблиц базы данных осуществлена по принципу выделения наиболее абстрактного и независимого базового объекта, максимально независимого от ПрО проектируемой ИС, который потом детализируется наследниками в соответствии со спецификой конкретной ПрО и объекта автоматизации. Такой подход обеспечивает возможность повторного применения требования при реализации других проектов, когда базовое понятие будет детализировано наследником, соответствующим новой ПрО, выдвигающей требование к создаваемой ИС, подобное уже реализованному в другой ИС требованию.

Рассмотрим иерархии фреймов, описывающих требования и образующих онтологию ПрО ИТ (рис. И.3, И.4).

Из рис. И.3, И.4 видно, что база данных, реализующая требования к учету проектов, структур этих проектов и описаний требований к этим проектам, описывается несколькими иерархиями фреймов, базовым из которых является Проект (Project) ИС. На рис И.3, соответствующем фрагменту базы данных, приведенному на рис. И.1, показано, что для описания требований к проекту ИС на уровне информации используется набор визуальных Диаграмм (Diagramm), которые являются Файлами (File), публикуемыми в системе управления требованиями.

В соответствии с сервисным подходом Проект представляется в виде набора ИТ-Сервисов (Service), связанных между собой потоками данных. Описание ИТ-сервисов с применением структурных диаграмм (например, DFD или IDEF0) позволяет отобразить декомпозицию ИТ-сервисов на отдельные Функциональные операции (Functional_operation) и взаимодействие ИТ-сервисов с внешней средой, описанной Внешними сущностями (External_entity). Поскольку и Сервис, и Функциональная операция, и Внешняя сущность являются элементами диаграмм, они наследуются от базового понятия Экземпляр (Instance). От этого понятия также наследуется фрейм-прототип Класс (Class), применяемый для описания типов сущностей и классов объектов в ER- и UML-диаграммах. Описание Классов детализируется посредством применения фреймов Атрибут (Attribute) и Метод (Method).

Приведенный набор фреймов-прототипов позволяет описать объекты-элементы структурных диаграмм при их обработке на уровне информации. Для описания визуальных моделей, отражающих процессы взаимодействия объектов ПрО, применяются фрейм Сообщение (Message) и унаследованный от него фрейм Операция (Operation). Фрейм Сообщение применяется для описания связей и информационных потоков между Экземплярами, например, в таких диаграммах, как IDEF0, IDEF3, IDEF1x и т.п. Фрейм Операция применяется для описания взаимодействия в UML-диаграммах, предполагающего вызов методов экземпляров классов, передачи им входных параметров и обработки выходных.

Фрагмент сети фреймов, приведенный на рис. И.4, описывает фрагмент базы данных ИТ, применяемый для работы с требованиями, представленными в виде данных (рис. И.2). Часть фреймов (Проект, Экземпляр, Сервис, Атрибут) на данном рисунке дублируется с рис. И.3. Фрейм-прототип Аналитическое требование (Analitical_requirement) отражает специфику работы с данным видом требований в ИТ. Поскольку источниками таких требований являются визуальные диаграммы, отображаемые в представление требований на уровне данных, и специфика работы с аналитическими требованиями предполагает их анализ и уточнение множеством пользователей, фрейм Аналитическое требование унаследован от фрейма Публикация. Таким образом, формирование и уточнение аналитических требований предполагает работу с Версиями

требований (Analitical_requirement_version) и Версиями атрибутов (Attribute_version). В ИТ требование публикуется и уточняется Пользователеми (User), которые могут выступать как Физические лица (Person_physical) или как представители Юридических лиц (Person_juridical), например, организации-Потребителя, т.е. Внешние лица (Outer_peson) по отношению к организации-Поставщику. При этом все классы лиц, участвующих в процессе формирования и управления требованиями к создаваемой ИС, обобщаются абстрактным фреймом Персона (Person).

В отличие от Аналитического требования, которое имеет множество версий, существует только на уровне информации или данных, и обладает свойствами, сходными с документом, Реализованное требование (Requirement) имеет физическую реализацию в виде элементов обеспечений ИС и является компонентом системы, то есть, Экземпляром (Instance), что обусловливает наследования Реализованного требования от данного фрейма. Поскольку версии типовых ИС в различных конфигурациях (то есть реализующие различные наборы требований) могут внедряться на нескольких объектах управления, для описания версии реализованной и внедренной ИС и набора реализуемых ею требований применяется фрейм Внедрение (Implementation), связанный с набором реализованных ИТ-Сервисов.

Приведенное описание сети фреймов иллюстрирует особенности интеллектуальной ИТ ускоренной разработки ИС. Для формирования сети фреймов, отображаемых впоследствии в элементы программного и информационного обеспечений ИТ-сервисов ИС, требуется применение приемов классического объектно-ориентированного анализа и проектирования (аналогично современным методологиям разработки ПО). Сформированная таким образом онтология ПрО пригодна для двустороннего отображения представления требований на уровне знаний в описания ПО и ИО ИС, как показано ниже в подразделе 5.6. Применение объектно-ориентированного анализа позволяет выявить объекты ПрО и связи между ними, в том числе сформировать иерархии классов объектов. Рассмотренные в подразделе 5.2 ИТ-услуги интеллектуальной ИТ ускоренной разработки ИС также предполагают расширение реализованных требований при их повторном использовании, то есть добавление новых уровней к существующим иерархиям объектов универсума требований к ИС.

Объектно-ориентированный анализ позволяет формировать иерархии классов ПрО неограниченно большой длины. Формально в современных языках программирования также нет ограничений на количество уровней наследования классов объектов. Однако на практике при проектировании и визуальном моделировании ПО не рекомендуется применять иерархии наследования, состоящие более чем из 5-6 уровней, так как это затрудняет восприятие диаграмм классов человеком [68, 147]. Кроме того, необходимость отображения сети фреймов в базу данных ИС также накладывает ограничение в

виде сложности реализации многоуровневых иерархий в виде наборов таблиц базы данных.

Вследствие этого наиболее сложной задачей при формировании онтологии ПрО в виде сети фреймов является выбор фреймов, являющихся корневыми элементами деревьев онтологии, и принятие решения о наследовании фреймов от существующих или о формировании нового корня дерева фреймов. При работе с деревьями требований с целью сравнения онтологий различных ПрО и принятия решения о расширении и повторном использовании требования в создаваемой ИС в рамках соответствующей ИТ-услуги наиболее сложной является задача оценки подобия объектов различных ПрО.

Для решения первой задачи введем следующее правило: если между понятиями A и B существует связь типа «1..∞», «0..∞», «1..m», «0..m», «m..m» или «∞..∞», то такая связь является признаком выделения понятия B как самостоятельного, а не наследуемого от A.

Выполнение данного правила обеспечит декомпозицию иерархии требований на отдельные узлы онтологий. Реализация таких онтологий целесообразна в виде отдельных сервисов и обеспечит возможность их двустороннего отображения в ПО и реляционные схемы данных, в которых в явном виде иерархические связи между таблицами отсутствуют.

Решение задачи сравнения ПрО проектируемой ИС с ПрО универсума библиотеки реализованных требований организации-разработчика предполагает осуществление следующих этапов, в результате которых формируется новая модель ПрО, основанная на библиотеке реализованных требований:

Этап 1. Анализ синонимов названий фреймов-элементов и их атрибутов для онтологий двух сравниваемых ПрО с целью максимального их приведения к общему словарю наименований.

Этап 2. Трансформация онтологий сравниваемых ПрО с использованием сформированного общего словаря наименований.

Этап 3. Пересечение двух полученных множеств объектов, выделение одинаковых объектов, характерных для обеих ПрО.

Этап 4. Сравнение состава атрибутов пар объектов. Принятие решения о повторном использовании существующих реализаций (в случае наличия всех требуемых атрибутов в существующей реализации) или о расширении существующей реализации путем создания нового объекта-наследника, детализирующего существующий объект (в противном случае). Включение полученных фреймов в итоговую модель ПрО.

Этап 5. Анализ объектов ПрО создаваемой ИС, не имеющих одноименных аналогов в универсуме реализованных требований. Попарное сравнение таких объектов со всеми объектами универсума реализованных требований с целью поиска объектов с максимальной степенью подобия. В случае успеха – принятие решений о повторном использовании существующих реализаций или

их расширении аналогично Этапу 4 в случае наличия объектов, подобных по составу атрибутов. Включение полученных фреймов в итоговую модель ПрО.

Этап 6. Включение оставшихся объектов ПрО в итоговую модель ПрО в виде новых иерархий фреймов.

Пример применения данных этапов приведен в подразделе 5.7, описывающем особенности практического применения интеллектуальной ИТ ускоренной разработки ИС.

Применение данных правил позволяет автоматизировать формирование советов аналитику в ходе создания и обработки сетей фреймов, а также автоматизировать формирование сети фреймов, описывающих онтологию ПрО с применением единой нотации в виде паттернов требований к ИС (включая повторно используемые паттерны, паттерны, требующие модификации, и новые паттерны, требующие разработки).

Повторное использование требований, включая их представление на уровне информации, данных, знаний, а также физическую реализацию в элементах программного и информационного обеспечений ИС требует формирование библиотеки паттернов требований. Описание ее реализации приведено на рис И.2 в виде набора таблиц, содержащих данные о требованиях, отображаемые в знания, классы ПО и таблицы БД. Библиотеку паттернов проектирования требований к ИС можно описать как многомерное хранилище данных, которое формируется представлениями, построенными на основе схем данных, приведенных на рис. И.1, И.2. Измерениями в этом хранилище будут являться: назначение паттерна, уровень применения паттерна, а также иерархии терминов ПрО. Фактами в этом хранилище будут являться описания паттернов конкретных требований к ИС, которые отражают особенности выполнения конкретных процессов обработки данных над конкретными структурами данных.

Поскольку требование, описанное в виде сети фреймов, может быть отображено в элементы его физической реализации, понятие паттерна требований на физическом уровне можно сформулировать следующим образом: паттерн требования – это совокупность сети фреймов, графических и текстовых описаний, а также многомерной витрины данных и иерархии классов ПО ИС. Как упоминалось ранее, каждый приведенный физический элемент паттерна, представленного в виде знаний, формально описывается с применением соответствующего математического аппарата, что является формальным признаком паттерна, как математически описанного приема или шаблона проектирования, позволяющего получить эффективный результат. Так, онтология элементов ПрО, реализуемых требованием, и структура элементов ПО ИС описываются в виде сети фреймов, дополненных каузальными сценариями, а структуры витрины данных описываются кортежными моделями. Применение таких паттернов упрощает также возможность использования визуальных моделей, облегчающих восприятие накопленных знаний в виде информации.

5.6 Особенности отображения онтологий предметной области в элементы информационного и программного обеспечений информационной системы

В качестве одного из результатов применения интеллектуальной ИТ ускоренной разработки ИС предполагается рассматривать осуществление взаимнооднозначного отображения сети фреймов, описывающей онтологию ПрО проектируемой ИС в виде паттернов требований к ИС, в элементы программного и информационного обеспечений ИС для последующего выполнения проектных работ по физической реализации системы. Теория фреймов является основой современных объектно-ориентированных языков программирования, и поэтому отображение паттернов требований, описанных в виде сети фреймов, в сущности базы данных и классы программного обеспечения разрабатываемой ИС сводится к решению задачи объектно-реляционного отображения.

Поскольку сеть фреймов и диаграмма классов основаны на применении общего математического аппарата, одинаковых типов связей и структур данных, отображение сети фреймов онтологии ПрО в диаграмму классов ПО осуществляется естественным образом: фреймы отображаются в классы с такими же названиями и наборами атрибутов и методов, связи между фреймами – в одноименные им связи генерализации и ассоциации классов.

Применительно к информационному обеспечению ИС основная сложность заключается в отсутствии в наиболее распространённой в настоящий момент реляционной модели данных иерархических связей и механизма наследования. Для реализации задачи объектно-реляционного отображения в настоящее время существует несколько типовых подходов, наиболее распространёнными из которых являются отображение каждого дочернего класса (включая наследуемые атрибуты) в отдельные таблицы базы данных, а также эмуляция механизма наследования за счет применения идентифицирующих связей между сильными (родительскими) и слабыми (дочерними) сущностями.

В рассматриваемой ИТ предлагается применять второй из описанных выше подходов. Взаимнооднозначное отображение терминов ПрО в таблицы базы данных ИС требует применения соответствующей методики построения. Согласно этой методике, база данных проектируется как набор взаимосвязанных витрин данных, реализованных в виде отдельных схем данных. В каждой из этих витрин таблицей фактов является корневой элемент конкретной иерархии фреймов, а консольным таблицам и таблицам измерений соответствуют уточняющие понятия ПрО. При этом наследуемые фреймы представляются как слабые сущности, зависящие от корневого понятия соответствующей иерархии фреймов онтологии ПрО, а каждая отдельная витрина данных представляет собой многомерный куб, который может быть свёрнут к одному универсальному отношению, содержащему все множество возможных атрибутов конкретной витрины данных.

В этом случае названия таблиц соответствуют фреймам, атрибуты таблиц – атрибутам фреймов, а присоединенные процедуры фреймов отображаются в хранимые процедуры и функции, а также в триггеры реляционной схемы данных. Иерархические связи отображаются в наборы сущностей, связанных между собой идентифицирующими связями, а связи-ассоциации – в ссылочные ограничения целостности данных, реализуемые с применением первичных и внешних ключей.

Рассмотрим особенности реализации наследования в реляционной базе данных более подробно. Для решения данной задачи предлагается выделение абстрактной обобщённой сильной сущности, которая соответствует корневому элементу конкретной иерархии фреймов и является универсальной для различных ПрО. Атрибуты такой сущности носят общий характер, а её применение предполагается при реализации требований, характерных для нескольких ИС. Для фреймов второго и выше уровней иерархии создаются слабые сущности, которые полностью зависят от более общей и универсальной сущности предыдущего уровня иерархии и дополняют её, поскольку содержат дополнительную информацию, характерную для более конкретного класса объектов ПрО ИС. Такая слабая сущность в качестве ключевого атрибута (PK) содержит внешний ключ (FK), реализующий связь «1:0..1» с обобщающим сильным типом сущности. При этом в случае, когда соответствующий ей фрейм имеет потомков, такая сущность в свою очередь является сильной для сущностей, соответствующих следующему уровню иерархии.

Для упрощения работы с базой данных ИС со стороны Поставщика ПО и Потребителя реализуются виртуальные таблицы – представления, которые содержат всю информацию об объекте ПрО, объединяя данные из нескольких таблиц (аналогично тому, как в ООП классы-потоки наследуют все атрибуты классов родителей). Использование представлений позволяет скрыть разветвлённость схемы данных от пользователей и обеспечить соответствие логического представления данных конкретной ПрО.

Несмотря на зависимость слабой сущности от сильной[15], распределение набора атрибутов одного реального объекта ПрО между несколькими таблицами, связанными идентифицирующими связями и отношениями «1:0..1» и созданными с целью реализации в реляционной БД иерархических связей, создает необходимость обеспечения целостности данных на уровне нескольких таких таблиц. Подобная целостность данных нехарактерна для классической ситуации размещения таких атрибутов в одной таблице (на этапе концептуального проектирования таблицы, имеющие связи «1:1», всегда объединяются в одну таблицу).

Вследствие этого именно необходимость отображения иерархии фреймов в реляционные схемы витрин данных обусловливает необходимость применения приведенного в предыдущем подразделе правила классификации фреймов как

[15] Такая зависимость выражается в том, что записи в слабой сущности не могут существовать без соответствующих записей в сильной сущности.

корневых самостоятельных понятий и в результате – представления сети фреймов в виде «низкорослого леса».

Поскольку ни целостность сущностей, ни ссылочная целостность в полной мере не обеспечивает решение данной задачи, предлагается использование триггеров. Триггеры реализуют сложные ограничения целостности данных, которые нереализуемы описательными ограничениями, устанавливаемыми при создании таблиц.

Можно выделить четыре основных типа триггеров, обеспечивающих целостность данных в обобщённой и уточняющих таблицах:

а) триггеры, обеспечивающие сюрьективное отображение данных в двух таблицах (обязательное наличие соответствующих записей в главной и зависимой таблицах, связь «1:1», поскольку наличие идентифицирующей связи обеспечивает только наличие связи «1:0..1»);

б) триггеры, обеспечивающие уникальность значений комбинаций атрибутов, расположенных в разных таблицах (аналог уникального ключа в одной таблице; для этой цели могут также применяться функциональные индексы);

в) триггеры, обеспечивающие контроль внесения данных в дочерние таблицы, когда сильная сущность связана с несколькими слабыми, а данные в слабых сущностях не должны пересекаться между собой (например, сильная сущность «человек», слабые – «преподаватель» и «студент»);

г) триггеры, обеспечивающие заполнение (аналог not null option) необязательных полей сильных сущностей, если это требуется для зависящих от них слабых сущностей (например, для абстрактной сильной сущности «человек» поле «контактный телефон» может быть необязательным, а для наследуемой от неё сущности «сотрудник» это же поле, физически расположенное в главной таблице, является обязательным).

Основные этапы построения сети фреймов в ходе анализа требований и ПрО создаваемой ИС, а также соответствующие им этапы отображения и реализации требований в описания элементов программного и информационного обеспечений ИС приведены в табл. 5.1.

Пример отображения иерархии фреймов ПрО в диаграмму классов ПО ИС приведен на рис. 5.2. Пример отображения сети фреймов в ER-диаграмму базы данных ИС приведен на рис. 5.3. Полученные в результате отображения иерархия классов и схема базы данных аналогичны фрагменту сети фреймов, приведенному на рис. И.4 и описывающему понятия физического и юридического лица. На рис. 5.2 представлена иерархия «Персона»-> «Юридическое лицо», «Физическое лицо» -> «Игрок» (PersonPhysicalLuk). При этом следует отметить:

а) факт повторного использования требования учета юридических и физических лиц, реализованных также в рассматриваемой ИТ;

Таблица 5.1 – Этапы формирования сети фреймов, диаграммы классов и схемы данных информационной системы

№	Моделирование ПрО	Проектирование ПО	Проектирование ИО
1	2	3	4
1	Выделение абстрактных понятий и их характеристик, универсальных для различных ПрО.	Выделение обобщенных (часто абстрактных) классов и наборов полей, универсальных для различных ПрО.	Выделение обобщенных сильных сущностей и наборов атрибутов, универсальных для различных ПрО.
2	Группировка выделенных понятий по признаку совместного использования. Выделение «подобластей».	Формирование пакетов классов.	Формирование отдельных схем (витрин) данных.
3	Формирование иерархий обобщенных понятий (выделение корневых фреймов, построение деревьев, состоящих из корня и ветвей).	1. Формирование иерархий абстрактных классов. 2. Формирование параметризуемых Generic-классов. 3. Формирование диаграмм абстрактных класссов, объединённых в пакеты.	1. Формирование обобщенных слабых сущностей, уточняющих сильные (связи 1: 0..1). 2. Формирование представлений, объединяющих обобщенные сильные и слабые физические сущности в логические. 3. Анализ функциональных зависимостей и определение необходимых ограничений целостности данных, в том числе реализуемых с применение триггеров. 4. Формирование ER-диаграмм.
4	Выделение понятий и их характеристик, характерных для конкретной ПрО.	Выделение классов и наборов полей, характерных для конкретной ПрО.	Выделение сущностей и наборов атрибутов, характерных для конкретной ПрО.

б) отличия в приведенных иерархиях фреймов на уровне объектов-листьев, которые отражают специфику конкретных ПрО: «Внешнее лицо» для рассматриваемой ИТ и «Игрок» для создаваемой ИС.

Продолжение табл. 5.1

1	2	3	4
5	Анализ возможности наследования фреймов выделенных понятий от универсальных, выделение новых универсальных понятий.	Анализ возможности наследования классов от существующих абстрактных классов.	Анализ возможности формирования слабых сущностей, уточняющих обобщенные сильные.
6	Формирование иерархий понятий, характерных для конкретной ПрО (добавление листьев к иерархиям понятий).	1. Формирование иерархий классов, характерных для конкретной ПрО. 2. Параметризация классов-листьев. 3. Формирование диаграмм классов, объеди-нённых в пакеты.	1. Формирование слабых сущностей-листьев, уточняющих обобщенные сущности. 2. Формирование представлений, объединяющих сильные и слабые физические сущности в логические. 3. Анализ функциональных зависимостей и задание ограничений целостности данных.
7	Расширение библиотеки паттернов новыми обобщенными понятиями, описанными фреймами.	Расширения ПО библиотеки паттернов новыми обобщенными классами.	Расширение ИО библиотеки паттернов новыми обобщенными сущностями.

На рис. 5.2 также показан фрагмент иерархии классов, реализующих паттерн проектирования ПО «Data Access Objects» (DAO), который предполагает создание иерархии специализированных классов для обеспечения взаимодействия ПО с базой данных, аналогичной иерархии бизнес-классов.

Характерной особенностью реализации DAO-слоя ПО является применение Generic-классов, параметризуемых на этапе выполнения (применительно к языку программирования Java). Возможно также применение паттерна проектирования ПО «Factory», применяемого для динамического создания объектов классов в зависимости от заданных условий. Поскольку при загрузке такими классами данных из базы осуществляется создание экземпляров классов ПО, в них нельзя применять обобщающие типы данных (в таком случае при загрузке данных об игроке, например, был бы создан объект «Физическое лицо», в котором отсутствуют атрибуты, специфичные для

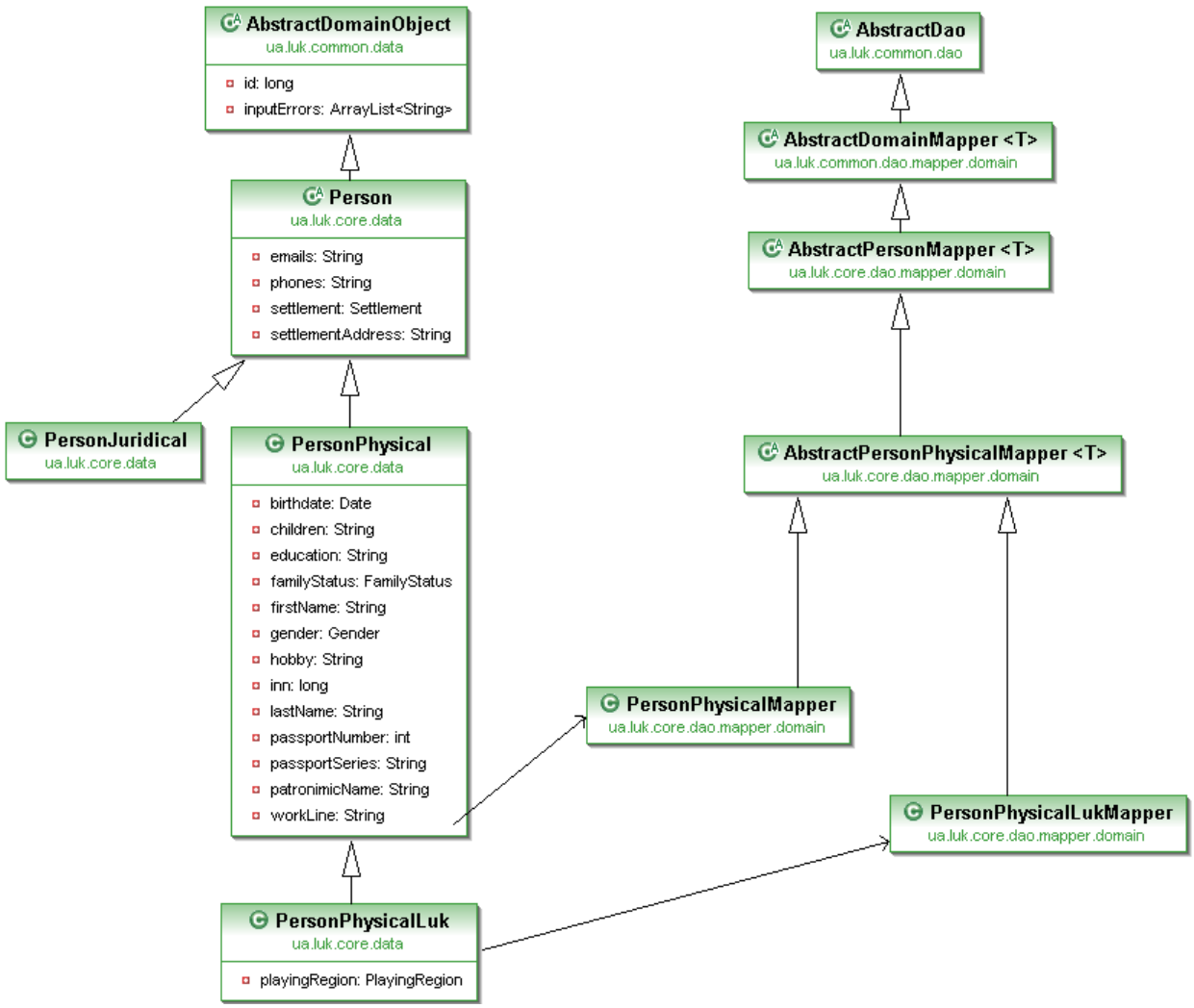

Рисунок 5.2 – Пример отображения сети фреймов в диаграмму классов программного обеспечения информационной системы

его наследников, и данные об игровом регионе в атрибут «playingRegion» загружены бы не были).

На рис. 5.3 приведена соответствующая схема данных, являющаяся результатом объектно-реляционного отображения сети фреймов в термины реляционной модели данных. Работу с этой схемой данных обеспечивают классы DAO-слоя ПО, показанные на рис. 5.2.

На данном рисунке также приведены представления, обеспечивающие формирование логических таблиц, соответствующих объектам реального мира, с полными наборами атрибутов (V_T_PERSON_PHYSICAL, V_T_PERSON_JURIDICAL, V_T_PERSON_PHYSICAL_LUK) на основе запросов к нескольким физическим таблицам. Эти логические таблицы отображают в базе данных иерархические связи между фреймами.

Как уже упоминалось выше, применение интеллектуальной ИТ ускоренной разработки ИС, необходимость работы с аналитическими

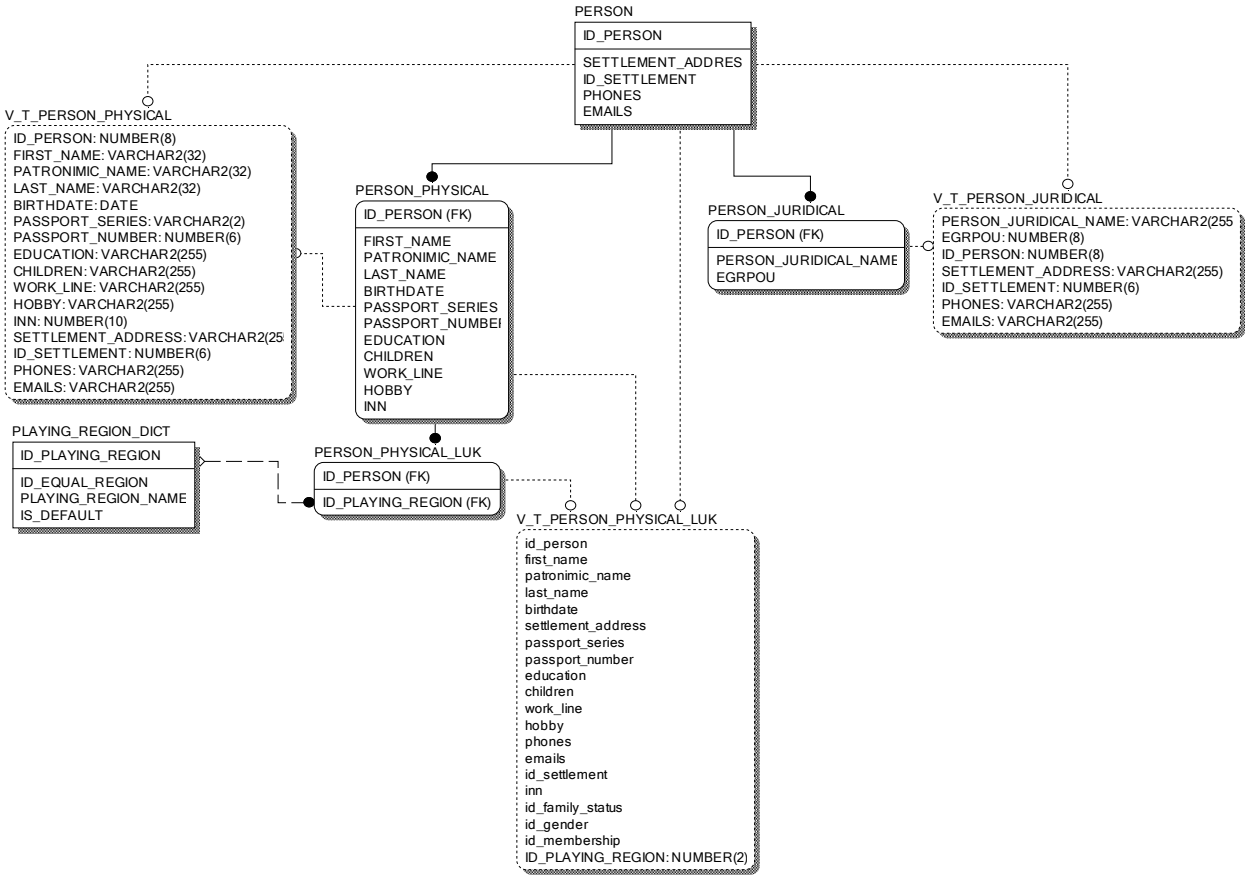

Рисунок 5.3 – Пример отображения сети фреймов в ER-диаграмму информационного обеспечения информационной системы

требованиями и формирования библиотеки реализованных требований, существенно увеличивают требования к ведению проектной документации. В наибольшей степени это касается наименования объектов программного и информационного обеспечений ИС. Целесообразной является разработка правил именования классов ПО, таблиц информационного обеспечения, а также их атрибутов, методов, процедур и других элементов.

Применение единой системы наименования элементов программного и информационного обеспечения необходимо для обеспечения двустороннего отображения диаграмм бизнес- и DAO-классов ПО в элементы информационного обеспечения ИС. Кроме того, это позволяет осуществлять автоматическую генерацию ограничений целостности при отображении сети фреймов, описывающих требования к ИС в элементы информационного обеспечения системы (автоматическую генерацию первичных и внешних ключей, а также описанных выше типовых триггеров на основе заранее заданных шаблонов). Для автоматической генерации ограничений целостности необходимо, чтобы правила именования были применены ко всем объектам базы данных независимо от того, созданы они вручную, или автоматически, например, с применением процедуры Forward Engeneering CASE-средств.

Пример соглашения об именовании объектов базы данных приведен в Приложении В.

Для контроля применения сформулированных и формализованных правил именования объектов возможна также реализация процедур автоматической проверки их соблюдения с применением словаря базы данных. Примеры процедур автоматической проверки правил именования объектов базы данных приведены в Приложении Г.

Шаблоны типовых триггеров, применяемых для реализации ограничений целостности, применительно к СУБД Oracle приведены в Приложении Д.

Примеры приведены применительно к проекту информационно-аналитической системы (ИАС) Всеукраинской общественной организации (ВОО) «Лига украинских клубов интеллектуальных игр» («ЛУК»), описание процессов создания которой представлено в подразделе 5.7.

Введение в модель фрейма (4.62) понятия «интерфейс фрейма» позволяет также реализовать отображение сети фреймов в элементы графического интерфейса пользователя. Требование на уровне знаний может быть описано в виде фрейма f, который может быть разделен на отдельные структуры данных, описываемые интерфейсами фрейма. Как и в объектно-ориентированном программировании, несколько различных фреймов, описывающих разные требования, могут реализовывать один и тот же интерфейс, то есть иметь общие структурные элементы. Применение интерфейсов фреймов позволяет применять в различных фреймах единые описания типовой структуры данных [148]. При этом класс или фрейм может одновременно реализовывать несколько интерфейсов. Кроме того, большинство современных языков объектно-ориентированного программирования не предоставляют возможность реализации множественного наследования, реализация которого вызывает неоднозначности (например, если в наследуемых классах существуют одноименные атрибуты). Применение интерфейсов позволяет осуществлять не только вертикальный анализ иерархической структуры классов или фреймов, но и горизонтальный анализ структурных элементов отдельных классов, что компенсирует отсутствие механизмов множественного наследования.

Комбинации двух указанных видов анализа структур фреймов позволяет осуществлять отображение фреймов, описывающих требования, не только в бизнес- и DAO-классы, но и в элементы ПО, описывающие интерфейс пользователя.

Разработка варианта интерфейса пользователя для заполнения данными составной части фрейма значительно менее трудоёмка, чем для всего элемента как единого целого, и позволяет сгруппировать поля для ввода логически связанных данных в виде отдельных визуальных компонентов, соответствующих либо интерфейсам классов, либо классам-родителям, структура которых является общей для нескольких классов-потомков. В программной реализации такие компоненты интерфейса пользователя для работы с документами могут быть представлены отдельными блоками-

контейнерами полей применительно как к web-, так и к GUI-интерфейсу ПО.

Если все фактографические поля класса описаны в реализуемых данным классом интерфейсах или наследуемых классах (структурных элементах фрейма el), которым в соответствие поставлены типовые визуальные компоненты, становится возможным автоматическое формирование интерфейса пользователя для работы с ним из типовых визуальных компонентов. При этом разработка отдельного варианта интерфейса пользователя для всего фрейма не требуется, что позволит снизить временны́е и трудозатраты на проектирование, избежать дублирования программного кода и обеспечить его повторное использование. Таким образом, будут снижены объем и количество ошибок в исходном коде ПО ИС, повышен коэффициент повторного использования стандартных программных компонентов [148].

Декомпозиция элемента документа упрощает адаптацию библиотеки реализованных требований к новым требованиям в ходе создания новых ИС или изменениям существующих требований в процессе их сопровождения, обеспечивает централизованное изменение требований к ИС совместно с компонентами интерфейса пользователя.

Очевидно, что одному структурному элементу фрейма el может соответствовать только один визуальный компонент pnl (сюръективное отображение $\mathrm{el} \to \mathrm{pnl}$), используемый для ввода и отображения данных. Поскольку, согласно (4.62), описание фрейма может содержать множество описаний интерфейсов этого фрейма, а иерархические связи между фреймами в сети фреймов позволяют определить множество наследуемых фреймов, данные соответствия позволяют в результате анализа структуры и иерархических связей фрейма сформировать интерфейс пользователя для заполнения фрейма фактографическими данными.

Поскольку понятие «интерфейс фрейма» введено в сеть фреймов из объектно-ориентированного программирования, особенности реализации интерфейсов классов в технологии объектно-ориентированного программирования действительны и применительно к представлению знаний в виде фреймов:

а) фрейм может содержать слоты, которые не принадлежат ни одному из реализуемых интерфейсов фрейма;

б) несколько интерфейсов фрейма могут регламентировать наличие во фрейме одного и того же свойства-слота;

в) свойство-слот, объявленное хотя бы в одном из реализуемых фреймом интерфейсов, один и только один раз должно быть объявлено во фрейме-прототипе.

Расширение модели сети фреймов понятием «интерфейс фрейма» (в объектно-ориентированном программировании реализация интерфейса класса, как правило, указывается при объявлении класса и описании его структуры) дополняет модель логического вывода с использованием иерархии фреймов.

Таким образом, модель формирования интерфейса пользователя ИС (PUI) для заполнения фрейма данными, основанная на анализе иерархических связей фрейма и его структуры, предлагается представить в следующем виде:

$$PUI(f) = \forall el \in f \Rightarrow \text{если } \exists pnl : el \to pnl = true \Rightarrow pnl \in UI, \qquad (5.1)$$

где f – фрейм, применяемый для описания требований в виде знаний;

el – элемент фрейма, являющийся либо интерфейсом, реализуемым фреймом (4.62), либо фреймом-прототипом, который он наследует;

pnl – визуальный компонент, элемент интерфейса пользователя ПО ИС;

$UI = \{pnl_1, ..., pnl_n\}$ – множество визуальных компонентов, составляющих интерфейс пользователя ИС для работы с бизнес-классом ПО, соответствующим фрейму f.

В модели (5.1) множество элементов фрейма el анализируется на предмет соответствия каждому из них визуального компонента интерфейса пользователя, используемого для отображения и внесения данных в бизнес-классы ПО. В случае наличия такого отображения ($el \to pnl$) визуальный компонент, соответствующий данному элементу фрейма, добавляется в интерфейс пользователя.

Данная модель может применяться для динамического формирования интерфейса пользователя ПО ИС в момент его отображения. Процедура динамического формирования интерфейса пользователя включает следующие этапы:

а) выделение множества элементов класса: интерфейсов, реализуемых фреймом, или наследуемых им фреймов-прототипов;

б) формирование множества визуальных компонентов для отображения в интерфейсе пользователя (5.1);

в) ранжирование множества визуальных компонентов по заданным приоритетам, определяющим порядок их расположения в интерфейсе пользователя;

г) отображение сформированного интерфейса пользователя ПО ИС для заполнения соответствующему фрейму бизнес-класса фактографическими данными.

В результате анализа элементов класса интерфейсов формируется список компонентов интерфейса пользователя, отображаемых в порядке, определяемом рангом или приоритетом, назначаемым каждому компоненту.

Таким образом, структура фрейма и реализующего этот фрейм программного класса однозначно определяют структуру интерфейса пользователя ИС. Этот интерфейс может формироваться как предварительно разработчиками, так и автоматически во время динамики выполнения исходных кодов ПО, и регламентирует набор фактографических данных, которые

отображаются пользователю, могут быть им модифицированы или внесены в бизнес-класс ПО, а соответственно и отображены в базе данных ИС.

Описанный подход к реализации объектно-реляционного отображения данных уменьшает производительность системы в целом за счёт множества дополнительных сущностей и связей «1:1», которые не рекомендуется использовать в схеме базы данных. Однако применение предлагаемого подхода сводит к минимуму контролируемую избыточность данных, сохранив преимущества применения реляционной модели для организации хранения данных в ИС. Кроме того, как было показано на примере формирования интерфейса пользователя, применение сети фреймов для описания требований к ИС позволяет также сохранить преимущества применения технологий объектно-ориентированного программирования. Таким образом, отображение сети фреймов в элементы программного и информационного обеспечений ИС обеспечивает реализацию предлагаемой интеллектуальной ИТ ускоренной разработки ИС, что позволяет облегчить Поставщику выполнение процессов создания последующих ИС на основе ранее разработанных систем, используемых в качестве прототипов. Тем самым, время разработки новых ИС значительно сокращается. Еще одним достоинством такого подхода является то, что в случае значительных изменений в БП, характерных для нескольких ИС, на этапе сопровождения данных систем в них достаточно перестроить только обобщённую часть системы, не выполняя индивидуальную доработку каждой системы.

5.7 Примеры использования интеллектуальной информационной технологии ускоренной разработки информационных систем

Для отражения особенностей практического применения предлагаемой интеллектуальной ИТ ускоренной разработки ИС целесообразно рассмотреть два следующих основных момента:

а) использование ранее выполненного проекта для выделения представлений требований к ИС и соответствующих этим требованиям проектных решений по программному и информационному обеспечениям ИС;

б) использование библиотеки реализованных требований к ИС в ходе создания новой ИС.

Для рассмотрения первого основного момента авторами была использована ИАС «Реестр ЛУК», которая создавалась по соглашению с ВОО «ЛУК» с использованием традиционных подходов. В качестве своеобразного архитектурного фреймворка в ходе создания ИАС «Реестр ЛУК» была использована методология, изложенная в ГОСТах группы 34 «Информационные технологии». Данная методология основана на каскадной модели жизненного цикла создания ИС. Описание особенностей ПрО и

основных пожеланий Потребителя, определяющих требования к ИС, были проанализированы в документе «Отчёт о проведённых работах на предпроектной стадии «Формирование требований» при разработке web-базированной информационно-аналитической системы «Реестр ЛУК», выдержки из которого приведены в Приложении Е.

В случае использования классического подхода к проектированию ИС с применением каскадной модели жизненного цикла системы, реляционная схема данных, описывающая ПрО ИАС «Реестр ЛУК», будет иметь вид, показанный на рис. И.5. На данном рисунке приведена реляционная схема данных, которая включает как таблицы, описывающие непосредственно ПрО ВОО «ЛУК», так и таблицы, необходимые для реализации нефункциональных требований к системе (разграничение прав доступа пользователей к системе, корректная обработка исключительных ситуаций при возникновении исключительных ситуаций в процессе работы с базой данных системы и т.п.).

Для реализации нефункциональных требований применяются следующие таблицы:

- ERROR_MESSAGES (классы ошибок, возникающих при работе с базой данных);
- DETAILED_ERROR_MESSAGES (детализированные описания ошибок в корреляции с объектами базы данных, при работе с которыми они могут возникнуть);
- MEMBER_WEB_USER (аккаунты пользователей ИАС, которые одновременно должны являться членами ВОО «ЛУК»);
- USER_PRIVILEGE (назначение пользователям системы привилегий доступа к web-страницам ИАС, реализующим те или иные её функции);
- WEB_PRIVILEGE (виды привилегий или ролей пользователей системы);
- WEB_PRIVILEGE_HIERARCHY (описание вложенности привилегий друг в друга);
- MENU_AVAILABLE_PAGE (описание прав доступа к web-страницам ИАС для различных привилегий пользователей);
- MENU_PAGE_JSP (описание физических web-страниц системы, права доступа к которым назначаются ролям пользователей);
- MENU_PAGE_MODE (описание функционального назначения физических web-страниц: просмотр, редактирование и т.п.);
- MENU_PAGE_COMPOSITION (включение физических web-страниц в состав логических страниц меню навигации web-ориентированной ИАС);
- MENU_PAGE (описание логических пунктов меню навигации системы);
- MENU_BLOCK (группировка пунктов меню).

Остальные таблицы реализуют функциональные требования в соответствии с функциональной структурой ИС, приведенной в Приложении Е.

Реинжиниринг ИАС «Реестр ЛУК», осуществлённый с применением интеллектуальной ИТ ускоренной разработки ИС, привел к модификации решений по видам обеспечений. Кроме того, дополнительно в состав ИАС «Реестр ЛУК»

включены функции CMS-системы, что позволяет отказаться от использования и интеграции с эксплуатируемой ранее в ВОО ЛУК CMS-системы Word Press. В результате решение функциональных задач ИАС «Реестр ЛУК» (см. Приложение Е) будет дополнено автоматической публикацией выходных документов в информационном web-портале ВОО ЛУК.

Модифицированная схема данных ИАС «Реестр ЛУК» приведена на рис. И.6. Исходная схема данных (рис. И.5) была расширена таблицами, необходимыми для реализации функций CMS-системы. Кроме того, в результате формирования сети фреймов, представляющих требования к системе на уровне знаний, и её отображения в информационное и программное обеспечение ИАС «Реестр ЛУК», исходная схема данных была разбита на несколько схем, реализованных в виде следующих витрин данных:

- APP_PERSON: таблицей фактов является PERSON (персона, лицо), производными от неё слабыми сущностями являются таблицы PERSON_JURIDICAL (юридическое лицо, организация) и PERSON_PHYSICAL (физическое лицо), таблицами измерений являются FAMILY_STATUS (семейный статус) и GENDER (пол);

- APP_CONTRACT: таблицей фактов является CONTRACT (контракт, факт занятия должности), таблицами измерений – POST (должность) и TYPE (тип должности);

- APP_GEOGRAPHY: таблицей фактов в зависимости от ситуации могут выступать SETTLEMENT (населённый пункт), REGION (регион, область), COUNTRY (страна, государство); таблица POPULATION_GROUP_DICT (категория населённости) всегда является таблицей измерений;

- APP_PUBLICATION: таблицей фактов являются PUBLICATION (публикация), VOTE (голос, оценка, рейтинг), COMMENTARY (комментарий), FILES (файл), PUBLICATION_FILE (прикреплённый к публикации файл);

- APP_PAYMENT: таблицей фактов является PAYMENT (платёж), таблицей измерений – RATE (тариф);

- APP_MEMBERSHIP: таблицами фактов являются MEMBERSHIP (членство) и MEMBER_PAYMENT (членский взнос, слабая сущность, производная от PAYMENT), таблицей измерений является REASON. При рассмотрении членских взносов таблица MEMBERSHIP выступает в роли таблицы измерения по отношению к таблице фактов MEMBER_PAYMENT;

- APP_TEAM: таблицей фактов является TEAM (команда, которая содержит иерархическую связь, отражающую преемственность команд), таблицей измерений – TEAM_MEMBERSHIP (состав команды как объединения игроков);

- APP_SESSION: таблицами фактов являются SEASON (сезон, период проведения турниров) и SESSION (сессия, чемпионат). При этом таблица SEASON выступает таблицей измерений по отношению к таблице SESSION, совместно с таблицами TYPE (тип чемпионата) и SESSION_STRUCTURE (состав чемпионата в виде календаря турниров);

- APP_MEETING: таблицами фактов являются MEETING (мероприятие, турнир) и MEETING_REQUEST (запрос на участие в турнире). При этом таблица MEETING выступает таблицей измерений по отношению к таблице MEETING_REQUEST. Таблицы TEAM_REQUEST (командная заявка) и PERSON_REQUEST (персональная заявка) являются слабыми сущностями, производными от MEETING_REQUEST, а таблица MEETING_TEAM_ MEMBERSHIP (заявленный состав команды) является таблицей измерений для таблицы TEAM_REQUEST;

- APP_LEAGUE: таблицами фактов являются LEAGUE (лига) и LEAGUE_MEMBERSHIP (членство команды в лиге), MEETING_LEAGUE (мероприятие определённой лиги, слабая сущность, производная от MEETING). При этом таблица LEAGUE выступает таблицей измерений по отношению к таблицам LEAGUE_MEMBERSHIP и MEETING_LEAGUE. Таблицы TYPE_DICT (тип лиги) и COMPLEXITY_LEVEL (уровень сложности) являются таблицами измерений;

- APP_USER: таблицей фактов является USERS (пользователь системы), таблицей измерений – USER_PRIVILEGE (привилегии пользователя в web-базированной ИАС, которым назначаются права доступа к отдельным физическим web-страницам, реализующим определённые функциональные операции ИТ-сервисов системы);

- APP_PRIVILEGE: таблицей фактов является PRIVILEGE (привилегия, роль), таблицами измерений – PRIVILEGE_GRANT (назначение вложенных ролей) и PRIVILEGE_AVAILABLE_PAGE (назначение прав доступа ролей к страницам web-приложения);

- APP_PAGE: таблицами фактов являются PAGE_LOGICAL (логическая страница), PAGE_PHYSICAL (физическая страница). Данные таблицы не являются производными общего понятия «Страница», поскольку логическая страница рассматривается в качестве пункта меню, а физическая в качестве адреса в структуре web-приложения, по которому осуществляется физический доступ к одной или несколькими функциональным операциям ИТ-сервиса. Таблицами измерения являются PAGE_MODE_DICT (тип функциональных операций, реализуемых страницей, например, редактирование или просмотр данных) и PAGE_COMPOSITION (объединение нескольких физических страниц в логическую страницу);

- APP_CORE: схема данных, применяемая для реализации нефункциональных требований к ИАС, описанных выше, для корректного отображения пользователю ошибок, возникающих при работе с БД (таблица фактов ERROR_MESSAGE и таблица измерений DETAILED_ERROR_ MESSAGE).

Для мониторинга использования дискового пространства сервера БД ИАС применяется таблица фактов AUDIT_SPACE.

Исключением являются таблицы схемы APP_LUK. Данная схема реализована не как витрина данных и содержит таблицы, которые либо

являются слабыми сущностями, дополняющими абстрактные таблицы фактов описанных выше витрин данных свойствами, характерными для ПрО ВОО «ЛУК», либо дополнительными таблицами измерений:

- PERSON_PHYSICAL_LUK (игрок, слабая сущность, производная от PERSON_PHYSICAL, расширена свойством «игровой регион»);

-PUBLICATION_LUK (публикация, слабая сущность, производная от PUBLICATION, расширена свойством «игровой регион»);

-USER_PRIVILEGE_LUK (привилегия/роль, слабая сущность, производная от PRIVILEG, расширена свойством «игровой регион», применительно к которому назначается привилегия);

- PLAYING_REGION (игровой регион) – таблица, дополняющая схему данных APP_GEOGRAPHY административно-территориальным делением ВОО «ЛУК»;

- AGE_CATEGORY (возрастная категория) таблица, дополняющая схему данных APP_PERSON возрастными категориями физических лиц;

- CONTRACT_LUK (контракт, слабая сущность, производная от CONTRACT, расширена свойствами, отражающими привязку должностей ВОО «ЛУК» к игровым регионам и наличие временных контрактов, действующих на время проведения одного чемпионата или отдельного турнира);

- RATE_LUK (тариф, слабая сущность, производная от RATE, расширена свойствами, отражающими зависимость тарифов членских и вступительных взносов ВОО «ЛУК» для физических и юридических лиц от возрастных категорий игроков и их принадлежности к лигам, а также неизменность размеров членских взносов в течение календарного года);

- TEAM_LUK (команда, слабая сущность, производная от TEAM, отражает привязку команд к населённым пунктам и игровым регионам, а также наличие команд, спонсируемых организациями);

- MEETING_LUK (турнир/игра, слабая сущность, производная от MEETING_LEAGUE, отражает лимит на количество участников мероприятия, а также размеры игровых взносов и признак зачета результатов мероприятия в итог чемпионата);

- MEETING_REQUEST_LUK (заявка, слабая сущность, производная от MEETING_REQUEST, отражает результаты участия в турнире и признак их влияния на общий зачет чемпионата);

- LEAGUE_LUK (лига, слабая сущность, производная от LEAGUE, отражает классификацию лиг ВОО «ЛУК» по игровым регионам, возрастным категориям, а также особенности переходов команд между лигами по результатам чемпионатов).

Как показано на рис. И.6, структура информационного обеспечения ИАС, реализующего определённый набор требований, состоит из множества витрин данных. В каждой из этих витрин присутствуют таблицы фактов и таблицы измерений, характеризующиеся общими ситуациями совместного их использования при решении функциональных задач системы. В роли таблиц

измерений могут также выступать таблицы других схем данных. Эта возможность отражает горизонтальные связи между терминами ПрО на уровне знаний и предполагает обмен данными связями между различными ИТ-сервисами и ИТ-услугами ИАС на уровне физической реализации системы.

При этом необходимо отметить следующие особенности:

а) реализация информационного обеспечения как набора витрин данных применяется независимо от типов требований – с применением данного подхода реализованы наборы как функциональных, так и нефункциональных требований к ИАС «Реестр ЛУК»;

б) корневые понятия сети фреймов и таблицы базовых витрин данных имеют максимально абстрактные названия и характеризуют абстрактные понятия, принадлежащие нескольким ПрО; уточнение данных понятий осуществляется при учете специфики конкретной ПрО путем их детализации в зависимых слабых сущностях ИО или наследуемых классах ПрО, что было показано применительно к схеме данных APP_LUK.

Кроме того, некоторые таблицы фактов (например, POST, FILE, RATE, SEASON на рис. И.6) могут быть выделены в отдельные витрины данных. Такое выделение целесообразно при наличии одной или нескольких уточняющих их таблиц измерений, которые могут быть выявлены, например, при разработке ИС для новых ПрО. Выделение же в отдельную витрину данных одной таблицы фактов в большинстве случаев является нецелесообразным с точки зрения удобства администрирования базы данных.

Фрагмент сети фреймов, отображенных в информационном обеспечении ИАС «Реестр ЛУК» (рис. И.6), который соответствует объектам ПрО ВОО «ЛУК», участвующим в формировании и оплате членских взносов, приведен на рис. И.7. Данный рисунок иллюстрирует практическое применение правила выделения корневых элементов сети фреймов, описанного в подразделе 5.5: между всеми фреймами первого уровня иерархии существуют связи 1:0..M и все фреймы-наследники имеют связи с фреймами-родителями 1:1 или 1:0..1. Следует отметить выделение в отдельное дерево понятия «Член организации», которое могло бы быть унаследовано от понятия «Персона». Такое выделение оправдано тем, что одна и та же персона может иметь длинную историю членства (например, быть членом организации в течение определенного периода, потом выйти из её состава, а потом вступить в организацию вновь), из-за чего между этими понятиями существует связь 1:0..M.

На данном рисунке также показаны зависимости между различными понятиями второго и третьего уровня реализации, которые реализуются за счет ассоциативных связей на уровне абстрактных понятий первого уровня иерархии.

Задание отношений ассоциации на максимально высоком уровне абстракции позволяет избежать их дублирования у фреймов-наследников, которые уточняют базовые понятия, что особенно важно при их отображении в таблицы базы данных. Фактически приведенная на рис. И.7 сеть фреймов

является гибридом, объединяющим понятия реляционной модели данных и объектно-ориентированного программирования, содержащим элементы и диаграммы классов, и ER-диаграммы витрины данных.

Применение данного подхода позволяет значительно облегчить отображение сети фреймов в описания элементов информационного и программного обеспечений ИС, поскольку данный подход основан на применении понятий и ограничений, характерных для обоих видов обеспечений системы. Следует также отметить, что если в сети фреймов ни корневой фрейм, ни его наследники не имеют ассоциативных горизонтальных связей с другими фреймами, факт отсутствия таких связей является критерием оценки «отмирания» и необходимости исключения данного требования из библиотеки требований.

Для рассмотрения второго основного момента был использован пример повторного использования требований, реализованных в ИАС «Реестр ЛУК», в создаваемой позднее ИС «Виртуальная доска объявлений». При разработке данной системы применялась популярная в настоящее время гибкая методология экстремального программирования, предполагающая активное вовлечение представителей Потребителя. Для данной ИС характерно отсутствие при определении представлений требований на уровне информации глубоко проработанного аналитиками и жестко регламентированного по структуре документа «Отчёт о проведённых работах на предпроектной стадии «Формирование требований», подкреплённого набором структурных диаграмм. Вместо него доступно только текстовое описание требований к создаваемой ИС, выполненное Потребителем, не являющимся специалистом в области проектирования ИС. Данный документ приведен в Приложении Ж и отражает формулируемые Потребителем требования к создаваемой ИС на уровне «как это вижу я».

В результате анализа требований к ИС «Виртуальная доска объявлений» и их пересечения с паттернами требований, реализованными в ИАС, выявлен набор требований, пригодных к повторному использованию. Схема данных, отражающая паттерны требований, готовых к повторному использованию, приведена на рис. И.8. К повторно используемым требованиям относится ряд нефункциональных требований, обеспечивающих реализацию администрирования системы, навигации и разграничения прав доступа к ИТ-сервисам и отдельным функциональным операциям (схемы APP_CORE, APP_USER, APP_PRIVILEGE, APP_PAGE), а также функциональные требования, реализующие функции CMS-системы в плане создания общедоступных публикаций (схема APP_PUBLICATION), и функции учета сведений о пользователях системы и их географии (схемы APP_PERSON и APP_GEOGRAPHY).

При реализации ИС «Виртуальная доска объявлений» (рис. И.9) схема APP_PERSON использована частично (только таблица PERSON, необходимая для учета контактных данных лиц, размещающих объявления).

Кроме того, с применением предложенной технологии повторного использования требований к ИС разработан ряд новых схем данных, реализованных в виде витрин данных, что обеспечивает возможность их применения при разработке следующих проектов:

– APP_ADVERTISEMENT: таблицами фактов являются ADVERTISEMENT (публикация-объявление, слабая сущность, производная от PUBLICATION), APPLIED_PROLONGATION (примененное продление показа объявления) и APPLIED_SELECTION (примененное продление выделения объявления). Таблица ADVERTISEMENT выступает в роли таблицы-измерения для APPLIED_PROLONGATION и APPLIED_SELECTION. Таблицами измерений являются PROLONGATION_TYPES (виды продления объявлений), ADVERTISEMENT_SELECTIONS (виды выделения объявлений) и ADVERTISEMENT_SELECTIONS_TYPES (категории выделения объявлений);

– APP_CURRENCY: таблицей фактов является UA_EXCHANGE_RATE (курс валюты по отношению к гривне, принятый в ИС «Виртуальная доска объявлений»), таблицей измерений является CURRENCY (валюта);

– APP_GOODS: таблицей фактов является GOODS (вид товара), таблицами измерений являются GOODS_CATEGORY (категория товара), MEASUREMENT (единицы измерения товара), COMPABILITY (совместимость товаров между собой, например, компьютерных комплектующих);

– APP_UNIT: таблицей фактов является UNIT (единица, изделие), таблица UNIT_COMPLECT (комплектующие) является слабой сущностью, зависимой от UNIT, расширяющей её атрибутами «серийный номер» и «гарантия». Таблица измерений – UNIT_STATUS (статус единицы товара);

– APP_OFFER: содержит таблицу фактов OFFER (предложение), выделена в отдельную схему данных, поскольку является одним из основных объектов учета в ИС «Виртуальная доска объявлений»;

– APP_SERVICE: содержит таблицу фактов SERVICE (услуга по диагностике), выделена в отдельную схему данных, поскольку является одним из основных объектов учета в ИС «Виртуальная доска объявлений».

Аналогично ИАС «Реестр ЛУК» в ИС «Виртуальная доска объявлений» в отдельную схему данных были выделены таблицы, уточняющие абстрактные таблицы фактов и вводящие новые таблицы измерений, характерные для ПрО данной системы:

– USER_DESK (пользователь, слабая сущность, производная от USERS, расширена свойствами, отражающими необходимость учета репутации пользователя Виртуальной доски объявлений и количества денег на его счете в системе);

– ADVERTISEMENT_DESK (объявление, слабая сущность, производная от ADVERTISEMENT, расширена свойством «дата продажи», необходимым для закрытия объявлений соответствующего типа);

– NOTED_ADVERTISEMENT (отмеченное объявление, слабая сущность, производная от ADVERTISEMENT и USER_DESK, отражает факт добавления объявления пользователем в свою «записную книжку»);

– ADVERTISEMENT_UNIT (объявление, слабая сущность, производная от ADVERTISEMENT и UNIT, отражает факт включения в объявление конкретного изделия и такой информации о нём, как цена, состояние и т.д.);

– ADVERTISEMENT_UNIT (объявление, слабая сущность, производная от ADVERTISEMENT_UNIT, расширяет её полем «гарантия», необходимым для комплектующих);

– SALE_TYPE (тип объявления о продаже – с фиксированной ценой или аукцион);

– PROVIDED_SERVICE (таблица, которая отражает факт заказа покупателем сервисных услуг по диагностике приобретаемого товара, их выполнение и оплату).

Приведенная схема данных демонстрирует практическое применение рассматриваемой интеллектуальной ИТ ускоренной разработки ИС к системам различных классов, разрабатываемых с применением различных методологий, моделей жизненного цикла, различной проектной документации для объектов различных типов и т.п., что свидетельствует об универсальности разработанной технологии.

Разработка организационной ИАС «Реестр ЛУК» с применением данной технологии позволила обеспечить повторное использование до 20% реализованных требований при создании самой интеллектуальной ИТ ускоренной разработки ИС (схемы APP_PERSON, APP_PUBLICATION, APP_USER) и до 40% при создании торговой ИС «Виртуальная доска объявлений» (схемы APP_CORE, APP_USER, APP_PRIVILEGE, APP_PAGE, APP_PUBLICATION, APP_PERSON и APP_GEOGRAPHY). Кроме того, разработка каждой новой системы обеспечивала расширение библиотеки паттернов реализованных требований, что позволит значительно повысить процент повторного использования требований при создании последующих ИС.

5.8 Оценка эффективности применения интеллектуальной информационной технологии ускоренной разработки информационных систем

Рассмотренная в предыдущих подразделах интеллектуальная ИТ ускоренной разработки ИС позволяет сформировать новый подход к решению важной проблемы – оценки затрат на создание ИС и, в частности их ПО. Данная проблема является особенно важной для организаций, занимающихся индустриальным созданием ПО ИС, чьи процессы можно охарактеризовать как находящиеся на втором и третьем уровнях зрелости соответствующей модели зрелости возможностей создания ПО (CMM) [75, 149].

В настоящее время признается, что оценить затраты на создание программного продукта в начале соответствующего ИТ-проекта на достаточно высоком уровне точности практически невозможно. Так, приемлемым уровнем точности оценки объема предполагаемого к созданию программного кода как основного аргумента модели COCOMO II (и, соответственно, затрат на создание программного продукта) на момент начала работ по оценке реализуемости считается диапазон 0,25..4 величины реального объема созданного программного кода [150]. При этом величина реального объема созданного программного кода становится известной только по окончании ИТ-проекта по созданию программного продукта, что делает оценку затрат на создание программного продукта в ходе инициации ИТ-проекта весьма приблизительной.

При рассмотрении других фаз ИТ-проекта по созданию программных продуктов наблюдаются следующие уровни точности оценивания объема предполагаемого к созданию программного кода (и, соответственно, затрат на создание программного продукта) [150]:

а) на момент начала работ по планированию ИТ-проекта – диапазон 0,5..2 величины реального объема созданного программного кода;

б) на момент начала работ по разработке программного продукта (эскизное проектирование) – диапазон 0,65..1,5 величины реального объема созданного программного кода;

в) на момент начала работ по детальной разработке программного продукта (техническое проектирование) – диапазон 0,75..1,25 величины реального объема созданного программного кода;

г) на момент начала работ по созданию и тестированию программного продукта (рабочее проектирование) – диапазон 0,85..1,15 величины реального объема созданного программного кода.

Данные особенности оценки затрат на создание программного продукта приобретают особое значение в ходе создания ИС. В соответствии с типовыми процессами создания ИС [50] решение о начале работ по созданию ИС принимается по результатам выявления потребностей будущих Потребителей, формирования на их основе множества требований к ИС и анализа полноты этих требований. Формализация и уточнение требований к ИС, а также синтез архитектуры создаваемой ИС осуществляется уже после того, как решение о создании ИС принимается Поставщиком и Потребителем. Однако, как показано выше, в ходе инициации ИТ-проекта получить достаточно точные оценки затрат на создание ИС весьма затруднительно.

Одна из основных трудностей расчета оценки затрат на создание программного продукта на ранних фазах ИТ-проекта связана с высокой неопределенностью оценки объема предполагаемого к созданию программного кода как основного аргумента модели COCOMO II (или аналогичных ей моделей). Существующие методы оценки данного показателя (метод

функциональных точек и его разновидности, метод объектных точек, метод ДеМарко, метод точек свойств, метод Wideband Delphi и т.п.) требуют для увеличения точности подсчета достоверных знаний об архитектуре ПО, которые на момент инициации ИТ-проекта по созданию ИС, как правило, отсутствуют. Использование же методов приближенного оценивания, основанных на знании общесистемных особенностей разрабатываемой ИС, дает весьма условные оценки.

Проблема оценки затрат на создание программного продукта на ранних фазах ИТ-проекта усложняется еще и возможностью повторного использования в этом проекте кода, разработанного в ранее выполненных ИТ-проектах. В этом случае необходимо уточнить объем предполагаемого к созданию программного кода с учетом предполагаемого объема повторно используемого кода и условий его повторного использования. Существующие методы оценивания объема кода (в частности, метод объектных точек) позволяют рассчитывать такое уточнение на ранних фазах ИТ-проекта весьма приблизительно.

Как показано в подразделе 3.4, интеллектуальная ИТ ускоренной разработки ИС, основанная на концепции представления требований к ИС (см. подраздел 3.1), ориентирована на возможность сокращения затрат на создание ИС за счет повторного использования компонентов информационного и программного обеспечений, соответствующих паттернам требований к ИС. Такой подход позволяет оценивать затраты на создание ИС по результатам анализа деревьев онтологий понятий предметной области, используемых для описания требований к ИС. Данный подход дает возможность также осуществить оценку степени повторного использования проектных решений за счет взаимно-однозначной связи между элементами деревьев онтологий и проектными решениями по видам обеспечений, реализующими соответствующие требования к ИС. При этом принятие решения о возможности повторного использования тех или иных фрагментов программного кода упрощается за счет анализа не самого кода, а понятий и терминов предметной области, описывающих требование к ИС, реализуемое данным фрагментом. Это, в свою очередь, позволяет представить оценку эффективности интеллектуальной ИТ ускоренной разработки ИС как величину сокращения трудозатрат на разработку ИС, которое обусловлено использованием данной ИТ.

Поскольку в основе интеллектуальной ИТ ускоренной разработки ИС лежат теория фреймов и объектно-ориентированный подход к созданию ИС, то для получения максимально релевантной оценки трудозатрат, необходимых на реализацию ИС, может быть применён метод объектных точек. Декларируемой областью применения метода объектных точек является оценка размера ПО в условиях использования взаимосвязанных прототипов с применением интегрированной CASE-среды ускоренной разработки (что в полной мере соответствует особенностям концепции представления требований к ИС и интеллектуальной ИТ ускоренной разработки ИС). Результаты сравнения методов объектных и функциональных точек показывают, что процедура

анализа трудозатрат на разработку ИС с учетом повторно используемого программного кода методом объектных точек характеризуется меньшей дисперсией, значительно меньшим временем, необходимым для получения оценки, и меньшей сложностью применения [151, 152].

Метод объектных точек базируется на процедуре, изложенной в [153], и данных о производительности, рассмотренных в [152].

В методе используются следующие определения терминов:

- NOP: новые объектные точки/новые точки привязки объектов (количество объектных точек с поправкой на повторное использование);
- srvr: количество расположенных на серверах (мэйнфреймы или их эквиваленты) таблиц данных, используемых в сочетании с экранами или отчетами;
- clnt: количество расположенных на машине клиента (персональной рабочей станции) таблиц данных, используемых в сочетании с экранами или отчетами;
- %reuse: процент повторно используемых экранов, отчетов и 3GL-модулей ранее разработанного приложения.

В качестве еще одного исходного определения отметим, что использование термина «объект» в методе объектных точек определяет как объекты экраны, отчеты и 3GL-модули. Это определение не связано с другими определениями термина «объект», основанными на таких признаках, как принадлежность к классу, наследование, инкапсуляция, передача сообщений и т.п.

Метод объектных точек содержит следующие этапы.

Этап 1. Оценить количество объектов исследуемого продукта как количество экранов, отчетов и 3GL-компонентов, которые будут входить в разрабатываемое приложение (при оценивании исходить из стандартного описания этих объектов в используемой интегрированной CASE-среде).

Этап 2. Классифицировать каждый экземпляр объекта по уровню сложности (простой, средний, высокий уровни) в зависимости от значений характерных размеров с использованием следующей схемы (табл. 5.2)

Этап 3. Взвесить результат классификации по следующей схеме (табл. 5.3). Веса отражают затраты, необходимые для реализации соответствующего экземпляра объекта указанного уровня сложности.

Этап 4. Для получения количества объектных точек ObjectPoints сложить значения весов всех взвешенных объектов.

Этап 5. Оценить процент повторного использования, которое ожидается в данном проекте, и рассчитать количество новых объектных точек по формуле

$$NOP = \frac{(ObjectPoints) \times (100 - \%reuse)}{100} . \qquad (5.2)$$

Этап 6. Определить коэффициент производительности PROD по следующей схеме (табл. 5.4).

Таблица 5.2 – Оценки уровня сложности экземпляров объектов в методе объектных точек

Для экранов				Для отчётов			
Количество используемых представлений (Views)	Источники в виде таблиц данных			Количество используемых секций	Источники в виде таблиц данных		
	Всего <4 (<2 srvr <3 clnt)	Всего <8 (2-3 srvr 3-5 clnt)	Всего 8+ (>3 srvr >5 clnt)		Всего <4 (<2 srvr <3 clnt)	Всего <8 (2-3 srvr 3-5 clnt)	Всего 8+ (>3 srvr >5 clnt)
<3	простой	простой	средний	0-1	простой	простой	средний
3-7	простой	средний	сложный	2-3	простой	средний	сложный
>8	средний	сложный	сложный	4+	средний	сложный	сложный

Таблица 5.3 – Значения весов уровней сложности объектов в методе объектных точек

Тип объекта	Уровень сложности		
	Простой	Средний	Сложный
Экранная форма	1	2	3
Отчёт	2	5	8
3GL-компонент			10

Таблица 5.4 – Значения коэффициента производительности PROD в методе объектных точек

Опыт и способности разработчиков, зрелость и возможности CASE	Очень низкий	Низкий	Нормальный	Высокий	Очень высокий
PROD	4	7	13	25	50

Этап 7. Рассчитать оценку количества человеко-месяцев, требуемых для реализации ИТ-проекта по разработке исследуемого продукта по формуле

$$PM = \frac{NOP}{PROD} \ . \qquad (5.3)$$

Однако специфика интеллектуальной ИТ ускоренной разработки ИС (в первую очередь особенности выбора корневых понятий онтологий предметной области) требует модификации метода объектных точек.

Прежде всего следует изменить определение объектной точки. С точки зрения интеллектуальной ИТ ускоренной разработки ИС, объектными точками следует считать отдельные деревья онтологии предметной области ИС, которые соответствуют отдельным таблицам схем данных типа «звезда» или «снежинка» в информационном обеспечении ИС (см. рис. 5.3) или совокупностям классов в программном обеспечении ИС (см. рис. 5.2)), реализующем бизнес-логику, экранные формы и отчеты ИТ-услуги.

Тогда показатель srvt будет определять количество используемых таблиц базы данных, расположенных на сервере (или серверах) ИС, характеризующих конкретную объектную точку и используемых в сочетании с экранными формами или отчетами. Показатель %reuse будет определять процент повторно используемых в ИС экранных форм, отчётов, таблиц базы данных и бизнес-классов ПО ранее разработанных ИТ-сервисов. При этом значение данного показателя следует определять на ранних стадиях ИТ-проекта создания ИС, исходя из оценки процента повторно используемых отдельных деревьев онтологии предметной области ИС как объектных точек.

Поскольку подавляющее большинство архитектур современных ИС (в том числе сервис-ориентированная архитектура) не предполагают хранение данных на компьютерах операторов и пользователей ИС, предлагаемая модификация метода объектных точек предполагает только оперирование суммарными показателями количества используемых таблиц (без их классификации на клиентские и серверные). В то же время сохранение значений и физического смысла базовых критериев позволяет применять проверенные на практике таблицы показателей базового метода (табл. 5.5).

Таблица 5.5 – Оценки уровня сложности экземпляров объектов в модифицированном методе объектных точек

Для экранов				Для отчётов			
Количество используемых представлений (view)	Количество таблиц базы данных, являющихся источниками данных			Количество структурных секций выходного документа	Количество таблиц базы данных, являющихся источниками данных		
	<4	<8	8+		<4	<8	8+
<3	простой	простой	средний	0-1	простой	простой	средний
3-7	простой	средний	сложный	2-3	простой	средний	сложный
>8	средний	сложный	сложный	4+	средний	сложный	сложный

Предлагаемая модификация метода объектных точек позволяет получить достаточно точные оценки объема трудозатрат на создание ИС уже в ходе инициации ИТ-проекта по созданию системы за счет использования для

подсчета моделей онтологий соответствующих понятий и терминов предметной области ИС, создаваемых на основе неформализованных описаний требований к ИС. При этом использование интеллектуальной ИТ ускоренной разработки ИС позволяет даже для создаваемой «с нуля» системы повторно использовать термины, основанные на них паттерны проектирования требований и, соответственно, проектные решения, заложенные в саму ИТ, что дает определенное сокращение трудозатрат. При этом рассчитываемые значения оценок трудозатрат на разработку ИС будут точнее оценок затрат, которые можно рассчитать с помощью других методов, применяемых на фазе инициации ИТ-проекта. Проиллюстрируем данные выводы на следующих примерах.

Рассмотрим вначале оценку трудозатрат на создание «с нуля» ИАС «Реестр ЛУК» с помощью упрощенного метода функциональных точек [154]. Данный метод предполагает использование эмпирического способа нахождения оценки функциональности в баллах в зависимости от классификации ИТ-проекта на фазе его инициации. В основе этого способа лежат три классификатора – масштаб проекта, характеристика пользователей объекта проектирования и тип объекта проектирования. Для ИАС ЛУК значения этих классификаторов равны следующим величинам (табл. 5.6).

Таблица 5.6 – Значения классификаторов для оценки функциональности в баллах

Масштаб проекта, С1	Пользователи объекта проектирования, С2	Тип объекта проектирования, С3
9 (приложение под заказ)	6 (внутрикорпоративное, распределенное использование)	6 (база данных)

Тогда оценка сложности ИТ-проекта создания ИАС в баллах функциональности может быть рассчитана по формуле [154]

$$FP = (C_1 + C_2 + C_3)^{2,35} \qquad (5.4)$$

и равняется с точностью до целого 1280 функциональных баллов.

Рассматривая в качестве основного языка программирования язык Java, получаем с помощью модели COCOMO[16] оценку трудозатрат на создание программного обеспечения ИАС «Реестр ЛУК», равной 173,25 человеко-месяца.

[16] Использование более точной модели COCOMO II не позволит получить серьезное уточнение в точности оценивания, поскольку объем повторно используемого кода предполагается равным 0, а определение значений драйверов затрат не дадут серьезных отклонений от результатов расчетов по модели COCOMO.

Теперь рассмотрим оценку трудозатрат на создание «с нуля» ИАС «Реестр ЛУК» с использованием в ходе соответствующего ИТ-проекта интеллектуальной ИТ ускоренной разработки ИС. При этом будем исходить из того, что разработанная ранее в соответствии с традиционным подходом версия данной системы не будет повторно использоваться.

В ходе выполнения Этапа 1 было установлено, что в ИАС «Реестр ЛУК» должно входить 14 объектных точек, представляющих собой деревья онтологий ПрО, реализуемые в виде витрин данных (рис. И.6). Для обработки и отображения данных по каждой из выделенных объектных точек используется в среднем по 3 экранных формы на 1 таблицу, а также 8 экранных форм для функций экспорта-импорта данных о составах команд в формат MS Excel (итого 197 экранных форм, физически реализуемых в виде JSP-страниц). Количество выходных документов, включенных в техническое задание на разработку ИАС «Реестр ЛУК», равняется 24.

В ходе выполнения Этапа 2 из 197 экранных форм 112 были классифицированы как простые, 46 – как средние, 39 – как сложные. Из 24 выходных документов 15 были классифицированы как средние, а 9 – как сложные отчёты. Простых отчётов в ИАС «Реестр ЛУК» не предполагается, поскольку многие экранные формы сами по себе являются простыми выходными документами, готовыми к печати. Все 14 архитектурных компонентов ИАС «Реестр ЛУК» были классифицированы как сложные.

В ходе выполнения Этапа 3 было проведено взвешивание результатов классификации объектных форм. Для экранных форм были получены следующие результаты: 112 ObjectPoints для простых форм, 92 ObjectPoints для средних, 117 ObjectPoints для сложных. Для отчетов были получены следующие результаты: 75 ObjectPoints для средних отчётов, 73 ObjectPoints для сложных. Результат взвешивания архитектурных компонентов равняется 140 ObjectPoints.

В ходе выполнения Этапа 4 суммарное количество объектных точек было определено как 609 ObjectPoints.

В ходе выполнения Этапа 5 был определен процент повторного использования деревьев онтологий. Поскольку ранее ИТ-проекты создания ИС с помощью рассматриваемой интеллектуальной ИТ ускоренной разработки ИС не выполнялись, то в качестве повторно используемых рассматривались деревья онтологий самой ИТ. Для ИАС «Реестр ЛУК» значение показателя %reuse равняется 20% (повторно используются онтологии ИТ, реализованные на рис. И.6 в виде фрагментов APP_PERSON, APP_PUBLICATION и APP_USER). Тогда количество новых объектных точек будет составлять $NOP = \frac{(609) \times (100 - 20)}{100} = 487{,}2$. Если же рассматривать применение модифицированного метода объектных точек без учета использования интеллектуальной ИТ ускоренной разработки ИС, то значение показателя

%reuse будет равно 0, а количество новых объектных точек будет составлять $NOP = \frac{(609)\times(100-0)}{100} = 609$.

В ходе выполнения Этапа 6 был осуществлен выбор значения коэффициента PROD. Поскольку проект разработки интеллектуальной ИТ ускоренной разработки ИС фактически являлся первым совместно выполненным проектом для участников команды разработчиков, а опыт участия в других проектах для половины членов команды невелик, на этапе разработки ИАС «Реестр ЛУК» средняя производительность команды разработчиков может быть оценена как нормальная (PROD=13).

В ходе выполнения Этапа 7 был проведен расчет количества человеко-месяцев, затрачиваемых на создание ИАС «Реестр ЛУК» с использованием интеллектуальной ИТ ускоренной разработки ИС. Результаты расчета составляют $PM = \frac{487,2}{13} \approx 37,48$ человеко-месяцев. Для сравнения результаты расчета количества человеко-месяцев, затрачиваемых на создание ИАС «Реестр ЛУК» без использования интеллектуальной ИТ ускоренной разработки ИС составляют $PM = \frac{609}{13} \approx 46,85$ человеко-месяцев.

В этом случае эффект от применения интеллектуальной ИТ ускоренной разработки ИС для ИТ-проекта создания ИАС «Реестр ЛУК» равняется $\Delta PM = \frac{609-487,2}{13} = 9,22$ человеко-месяца.

Результаты сравнения оценок трудозатрат на создание ИАС «Реестр ЛУК» с реальными трудозатратами приведены в табл. 5.7. Значение реальных трудозатрат было получено после завершения проектных работ по созданию ИАС «Реестр ЛУК» с использованием интеллектуальной ИТ ускоренной разработки ИС.

Таблица 5.7 – Оценочные и действительное значения трудозатрат на создание информационно-аналитической системы «Реестр ЛУК», человеко-месяцы

Упрощенный метод функциональных точек	Метод объектных точек без использования интеллектуальной ИТ ускоренной разработки ИС	Метод объектных точек с использованием интеллектуальной ИТ ускоренной разработки ИС	Действительное значение трудозатрат, определенное после завершения ИТ-проекта
173,25	46,85	37,48	26,58

Теперь рассмотрим эффект от применения интеллектуальной ИТ ускоренной разработки ИС для ИТ-проекта создания ИС «Виртуальная доска

объявлений». Как показано в подразделе 5.7, при создании данной ИС были повторно использованы паттерны проектирования требований к ИС, основанные на онтологиях предметной области, реализованных в ходе создания ИАС «Реестр ЛУК» (а именно деревья онтологий, реализованные в виде фрагментов схем APP_CORE, APP_USER, APP_PRIVILEGE, APP_PAGE, APP_PUBLICATION, APP_PERSON и APP_GEOGRAPHY, показанных на рис. И.8). В целом количество повторно используемых деревьев онтологий и, соответственно, витрин базы данных равно 7. «С нуля» были разработаны 7 новых деревьев онтологий, отраженных в соответствующие витрины базы данных (см. рис. И.9). При этом для создания обеспечивающей части ИС «Виртуальная доска объявлений» применялись те же языки и платформы разработки, что и при создании ИАС «Реестр ЛУК» (Java7 SE, JSP 2.2 и Oracle XE 11g R2).

Рассмотрим оценку трудозатрат на создание ИС «Виртуальная доска объявлений» с использованием в ходе соответствующего ИТ-проекта интеллектуальной ИТ ускоренной разработки ИС.

В ходе выполнения Этапа 1 было установлено, что в ИС «Виртуальная доска объявлений» также должно входить 14 объектных точек, представляющих собой деревья онтологий предметной области, реализуемые в виде витрин данных (рис. И.9), для работы с которыми также используется в среднем по 3 экранных формы на 1 таблицу базы данных. Итого ИС «Виртуальная доска объявлений» может быть на данном этапе оценена как 144 экранные формы, физически реализованные в виде JSP-страниц. Количество выходных документов, включенных в техническое задание на разработку ИС, равняется 12.

В ходе выполнения Этапа 2 из 144 экранных форм 82 были классифицированы как простые, 38 – как средние, 24 – как сложные, а из 12 выходных документов 8 были классифицированы как средние и 4 – как сложные отчёты. Все 14 архитектурных компонентов ИС были классифицированы как сложные. Функциональность некоторых повторно используемых компонентов используется не в полном объеме, что следует учесть при экспертной оценке значения процента повторного использования кода %reuse.

В ходе выполнения Этапа 3 было проведено взвешивание результатов классификации объектов. Для экранных форм были получены следующие результаты: 82 ObjectPoints для простых экранных форм, 76 ObjectPoints для средних экранных форм, 72 ObjectPoints для сложных экранных форм. Для отчетов были получены следующие результаты: 40 ObjectPoints для средних отчётов, 32 ObjectPoints для сложных отчётов. Результат взвешивания архитектурных компонентов равняется 140 ObjectPoints.

В ходе выполнения Этапа 4 суммарное количество объектных точек было определено как 442 ObjectPoints.

В ходе выполнения Этапа 5 был определен процент повторного использования деревьев онтологий. Как было отмечено выше, для повторного использования предлагаются 7 деревьев онтологий из 14 реализуемых, однако функциональность, реализуемая некоторыми из этих 7 деревьев, используется в ИС «Виртуальная доска объявлений» не в полном объеме. Поэтому в ходе экспертизы соответствующего ИТ-проекта было установлено, что значение показателя %reuse равняется 40. Тогда количество новых объектных точек для ИС «Виртуальная доска объявлений» составляет $NOP = \frac{(442) \times (100 - 40)}{100} = 265{,}2$.

В ходе выполнения Этапа 6 был осуществлен выбор значения коэффициента PROD. Поскольку ИТ-проект по созданию ИС «Виртуальная доска объявлений» был реализован непосредственно сразу после окончания разработки ИАС «Реестр ЛУК», уровень зрелости и опыта команды разработчиков повысился весьма незначительно. Поэтому примем PROD=13,5.

В ходе выполнения Этапа 7 был проведен расчет количества человеко-месяцев, затрачиваемых на создание ИС «Виртуальная доска объявлений» с использованием интеллектуальной ИТ ускоренной разработки ИС. Результаты расчета составляют $PM = \frac{265{,}2}{13{,}5} = 19{,}64$ человеко-месяцев.

Следует отметить, что использование модифицированного метода объектных точек в ходе инициации ИТ-проекта создания ИС «Виртуальная доска объявлений» без использования рассматриваемой интеллектуальной ИТ ускоренной разработки ИС требует от аналитика, проводящего расчет, хорошего знания о возможности повторного использования паттернов проектирования требований к ранее реализованной ИАС «Реестр ЛУК». С ростом успешно завершенных ИТ-проектов знания о допустимых для использования паттернов проектирования требований к ИС сохраняются и обрабатываются в интеллектуальной ИТ ускоренной разработки ИС (в отличие от памяти аналитика, проводящего расчеты на фазе инициации нового ИТ-проекта). На практике значение показателя %reuse определяется аналитиком как минимально возможное для снижения риска ошибки при повторном использовании проектных решений ранее выполненных ИТ-проектов. При этом на оценке аналитиком значения показателя %reuse будет также сказываться сложность сравнения аналитиком описаний объективно схожих друг с другом процессов в различных предметных областях создаваемых ИС. Не менее важное влияние на подобный подход к оценке будет также оказывать высокая текучесть кадрового состава, характерная для большинства ИТ-предприятий Украины.

Исходя из сказанного, целесообразно предположить, что в большинстве случаев при инициации очередного ИТ-проекта создания ИС, предметная область которой будет значительно отличаться от предметных областей ранее

созданных ИС, аналитик будет принимать значение показателя %reuse равным 0. Тогда полученная оценка трудозатрат будет хотя и завышенной, но все же достаточно близкой к ожидаемому реальному значению трудозатрат.

В этом случае эффект от применения интеллектуальной ИТ ускоренной разработки ИС для ИТ-проекта создания ИС «Виртуальная доска объявлений» равняется $\Delta PM = \frac{442 - 265{,}2}{13{,}5} = 13{,}1$ человеко-месяцев.

Таким образом, использование предлагаемой интеллектуальной ИТ ускоренной разработки ИС позволяет в ходе инициации ИТ-проекта создания ИС и, соответственно, в условиях полной неопределенности архитектуры создаваемой системы получить достаточно точную оценку трудозатрат на реализацию данного ИТ-проекта. При этом точность получаемой оценки (при использовании модифицированного метода объектных точек) укладывается в приведенный выше диапазон значений оценки трудозатрат на момент начала работ по планированию данного ИТ-проекта. Следует отметить, что оценка трудозатрат в ходе планирования ИТ-проекта формируется в условиях полной определенности архитектуры создаваемой системы и некоторого риска прямого оценивания объема создаваемого и повторно используемого кода.

Помимо повышения точности оценивания трудозатрат не менее важным эффектом от применения рассматриваемой интеллектуальной ИТ ускоренной разработки ИС следует считать сокращение трудозатрат на создание ИС за счет повторного использования требований к ИС и реализующих эти требования ИТ-сервисов. При этом величина эффекта от применения данной ИТ, как показано выше, сравнима с общими оценками трудозатрат на создание ИС. Следовательно, и экономический эффект от использования интеллектуальной ИТ ускоренной разработки ИС следует ожидать в аналогичном соотношении.

5.9 Выводы

Рассмотренная в данном разделе интеллектуальная ИТ ускоренной разработки ИС обеспечивает формирование архитектурного стиля системы, особенности которого были описаны в предыдущих подразделах. К практическому воплощению этих особенностей относятся: применение формализованных правил именования объектов информационного и программного обеспечения системы (Приложение В), реализация однозначного двустороннего отображения элементов описания архитектуры ИС в её информационное и программное обеспечения, разработка структуры базы данных как набора витрин данных, представление конфигурации ИС как сборки отдельных ИТ-сервисов и т.д.

Рассмотренная ИТ оперирует формальным описанием архитектуры (Architecture Description) структурных элементов ИС (ИТ-услуг и ИТ-сервисов),

полученным в результате анализа требований и выполненным в соответствии со стандартом ISO/IEC/IEEE 42010. Для описания требований, отображаемых в элементы создаваемой ИС, рассмотренная ИТ регламентирует использование моделей представления требований на уровне информации (визуальные структурные и объектные диаграммы), данных (содержимое схемы данных) и знаний (сеть фреймов). Кроме того, ИТ регламентирует правила формирования предложенных моделей и процедуры их отображения друг в друга, а также в элементы информационного и программного обеспечения ИС (то есть операции с моделями, применяемыми для описания элементов архитектуры ИС).

Исходя из сказанного выше, следует считать справедливым следующее утверждение: архитектурный стиль, регламентируемый рассмотренной ИТ, согласно стандарту ISO/IEC/IEEE 42010 является практическим воплощением формальной части АФУР ИС (см. подразделы 1.3 и 4.1):

– приведены описания классов ИС, при разработке которых применима данная ИТ и основанный на ней фреймворк (System-of-interest);

– приведены описания сторон, заинтересованных в разработке ИС (Stakeholder);

– сформулированы описания назначения требований к разрабатываемой ИС и функциональных задач, решаемых с её помощью (Concern);

– рассмотрены различные группы требований к системе, отражающие точки зрения множества групп пользователей и разработчиков систем (Architecture of Viewpoints), представленные на уровне информации, данных и знаний (Model Kind);

– в формализованное представление требований к ИС включены их описания (Architecture View), представленные в различных формах и нотациях (Architecture Model);

– регламентирован набор возможных связей между требованиями к ИС (Correspondence Rule), в процессе формирования конфигурации ИС между требованиями устанавливаются соответствующие взаимосвязи (Correspondence);

– применение предложенных ИТ и фреймворка обосновано (Architecture Rationale) и обеспечивает синтез архитектурных решений для конкретной системы (Architecture), обеспечивающей решение заданного множества задач.

Приведенное описание отражает согласованность рассмотренного в разделе 4 АФУР ИС с положениями концептуальной модели стандарта ISO/IEC/IEEE 42010, а также включает характеристики проблем, решаемых посредством применения данного фреймворка, сторон, заинтересованных в его применении, условий применимости и т.д. При этом, в соответствии с требованиями данного стандарта для архитектурных фреймворков, интеллектуальная ИТ ускоренной разработки ИС предполагает применение описаний различных архитектурных элементов, то есть требований к ИС (различных точек зрения участников проектирования и пользователей системы), представленных в любом виде (информация, данные, знания),

созданных с применением любых языков (моделей) и средств моделирования (платформ, CASE-средств и т.д.).

Стандарт ISO/IEC/IEEE 42010 также декларирует, что составной частью архитектурного фреймворка являются правила соответствий, обеспечивающие интеграцию различных точек зрения на архитектуру и функциональность проектируемой системы в единое целое. Следует отметить, что стандарт только выдвигает требования наличия такого компонента, но не регламентирует способы его реализации, а существующие фреймворки реализуют его в виде набора рекомендаций. В разделах 4 и 5 этот компонент АФУР ИС описан формально и реализован на практике.

Таким образом, полученные в разделе 4 модели АФУР и реализованные на их основе в разделе 5 решения интеллектуальной ИТ ускоренной разработки ИС удовлетворяют всем формальным признакам архитектурного фреймворка и в полной мере соответствуют стандарту ISO/IEC/IEEE 42010.

Рассмотренная ИТ доведена до практической реализации и апробирована при разработке двух ИС различного назначения. Данная технология является самоописательной, что было подтверждено её применением при создании решений собственной обеспечивающей части.

Тем не менее, данная ИТ имеет следующие недостатки:

- усложнение стадии анализа требований к ИС;
- повышение требований к качеству ведения проектной документации;
- введение дополнительных иерархий в модели ПрО, диаграммы классов и схемы данных;
- повышение сложности моделирования ИС;
- снижение производительности информационного обеспечения ИС в целом за счет дополнительных операций соединения при выборке данных и применения триггеров.

При этом преимуществами использования рассмотренной ИТ являются:

- формальное определение возможности повторного использования библиотеки существующих компонентов;
- снижение времени создания новых ИС за счет наращивания, а не реинжиниринга используемого прототипа;
- аккумуляция знаний о различных ПрО в универсальном прототипе, их многократное использование в различных ИС;
- снижение сложности сопровождения нескольких систем, разработанных на базе одного прототипа.

ЗАКЛЮЧЕНИЕ

Подводя итоги, необходимо особо отметить следующие положения, определяющие основные результаты исследования.

1. Проблемы, связанные с формированием и анализом требований, выдвигаемых к ИС различного назначения, по-прежнему не имеют комплексного решения. Так, например, решение задачи создания архитектуры ИС на основе результатов формирования и анализа требований к ИС до сих пор носит интуитивный характер и является одним из основных факторов, отрицательно влияющих на ИТ-проект создания ИС. Вследствие этого процессы создания ИС, непосредственно работающие с требованиями и формирующие архитектуру ИС (или ее программной реализации), в настоящее время рассматриваются как одно из наиболее серьезных «узких мест» таких ИТ-проектов, а исследования в данной области считаются одними из наиболее актуальных в компьютерных науках.

2. Большинство современных ИС и, соответственно, их архитектуры созданы в соответствии с сервисным подходом к созданию ИС. Однако определения основных терминов, используемых для описания ИС в соответствии с данным подходом, по-прежнему неоднозначны. В монографии предлагается уточнение понятий «ИТ-услуга» и «ИТ-сервис», определения архитектуры ИС на различных уровнях представления, а также скорректированные описания основных ролей участников ИТ-проекта создания, внедрения и эксплуатации ИС.

3. Исходя из предложенного представления ИС как совокупности ИТ-услуг, в монографии предлагается рассматривать основные взаимодействия Поставщика и Потребителя ИТ-услуг как частный случай концепции CRM, учитывающий специфику жизненного цикла ИС. На основании данной концепции сформулированы определения и предложены обобщенные формализованные описания глобальных целей Поставщика и Потребителя в процессе проектирования архитектуры ИТ-услуг ИС ((2.18)-(2.21)) и процессе проектирования архитектуры ИТ-сервисов ИС ((2.22)-(2.25)), а также определено обобщенное представление взаимодействия Поставщика и Потребителя в процессе проектирования архитектуры ИС как бескоалиционная игра двух игроков (2.26). Предлагаемые формализованные описания позволяют определить подход к поиску конкретных решений проблем формирования и анализа требований, а также проектирования архитектуры ИС.

4. В качестве основы решения проблем, возникающих в ходе непосредственной работы с требованиями и проектирования архитектуры ИС, в монографии рассматривается предложенная концепция представления требований к ИС. Данная концепция представляет собой набор следующих положений:

а) отказ от рассмотрения только множества сформулированных требований к ИС и изначальное представление требований к ИС как элементов универсума, включающего в себя как известные, так и неизвестные Поставщику, Потребителю или им обоим требования к ИС, а также методы формирования этих требований;

б) изначальное многообразие представлений требований к ИС в виде данных, информации и знаний;

в) процессный подход к описанию требований, определяющий минимальную процессную атрибутивную модель требования к ИС;

г) подход к управлению требованиями к ИС, основанный на изложенном в подразделе 1.5 основном принципе управления требованиями.

5. На основе изложенной концепции представления требований к ИС в монографии разработаны модели невыявленных требований (3.10), частично выявленных требований (3.14) и (3.15), а также групп требований к ИС (3.16), (3.20) – (3.25). Данные модели были использованы для разработки:

а) модели сформулированных требований (3.26), которая с прикладной точки зрения является формализованным описанием набора допустимых для реализации ИТ формирования и анализа требований к ИС;

б) модели универсума требований (3.27), которая с прикладной точки зрения является формализованным описанием фрагмента допустимых для реализации методологий проектирования ИС, рассматривающего возможные модели и методы формирования и анализа требований к ИС.

6. Для определения основных способов практической реализации моделей (3.26) и (3.27) в монографии рассмотрены особенности одноместного ковариантного функтора как основного способа преобразования представлений групп требований друг в друга. Предложено формализованное описание одноместного ковариантного функтора (3.38) и выделены два основных способа его реализации.

7. По результатам анализа предложенных способов реализации одноместного ковариантного функтора была выделена проблема повторного использования в ходе выполнения проекта создания новой ИС требований к ИС, сформулированных в предыдущих ИТ-проектах. В качестве решения этой проблемы в монографии предложено определение паттерна проектирования требования к ИС, математическая модель (3.39), описывающая подобные паттерны, а также модель (3.44) подмножества интеллектуальных ИТ формирования и анализа требований к ИС, которые могут быть реализованы на основе концепции выявления и использования знаний о предметной области и ИС в виде паттернов. Применение ИТ из данного подмножества позволяет вместо повторного использования отдельных представлений требований к ИС осуществлять повторное использование знаний, выявленных в ходе формирования и анализа требований, выдвинутых к различным ранее разрабатывавшимся ИС.

8. В ходе реализации разработанных моделей в монографии предложено определение понятия «архитектурный фреймворк быстрой разработки

информационной системы» и определено место АФБР ИС в процессах, непосредственно работающих с требованиями, а также в процессе проектирования архитектуры системы. Разработана модель АФБР ИС (4.2), в которой набор видов моделей, используемых фреймворком, представляет собой совокупность структурных паттернов проектирования требований к ИС. Данная модель устанавливает общие особенности и ограничения практик создания, интерпретации, анализа и использования AD ИС на уровне требований, выдвигаемых к данной ИС, в процессах предпроектного обследования и синтеза единого целостного AD ИС в виде ее ФС.

9. Для успешной реализации структурных паттернов как основополагающего компонента модели формальной части АФБР ИС в монографии были определены основные способы представления требований к ИС на уровнях данных, информации и знаний, а также сформулированы основные способы представления паттернов проектирования требований к ИС на соответствующих уровнях. Предложена обобщенная модель сформулированного требования к ИС в виде категории (4.3), на основе которой разработаны:

а) модели структурных паттернов (4.8) – (4.13) и поведенческих паттернов (4.17), (4.21), (4.23) – (4.33), определяющие синтаксис и семантику атрибутивных моделей требований к ИС на уровне данных;

б) модели структурных паттернов (4.35) – (4.39) и поведенческих паттернов (4.42) – (4.58), определяющие синтаксис и семантику атрибутивных моделей требований к ИС на уровне информации;

в) модели структурных паттернов (4.64) – (4.67) и поведенческих паттернов (4.70)-(4.86), определяющие синтаксис и семантику описаний знаний, добытых из публикаций требований к ИС.

10. Основываясь на полученных результатах, в монографии рассматривается основное определение термина «интеллектуальная ИТ быстрой разработки ИС» и описание ее основных ИТ-услуг. Разработана схема взаимодействия в рамках данной ИТ основных представлений требований к ИС на уровнях информации, данных и знаний. Определены основные достоинства и недостатки данной ИТ. Кратко рассмотрены основные особенности обработки представлений требований к ИС на уровне информации, данных и знаний, выполняемой в рамках интеллектуальной ИТ быстрой разработки ИС, рассмотрены основные схемы данных, используемые для хранения описаний требований к ИС на различных уровнях представления.

11. Рассмотрены особенности использования разработанной интеллектуальной ИТ быстрой разработки ИС в ходе создания ИАС «Реестр ЛУК» и ИС «Виртуальная доска объявлений». Наглядно показана возможность повторного использования формальных описаний требований к ИС и соответствующих им проектных решений по информационному и программному обеспечениям в ходе создания новой ИС. Осуществлены оценки трудозатрат на создание этих ИС. Проведено оценивание эффекта от использования разработанной ИТ, который выражается в сокращении трудозатрат на создание ИС за счет

повторного использования требований к ИС и соответствующих проектных решений. Для ИАС «Реестр ЛУК» экономия трудозатрат за счет использования данной ИТ для повторного использования требований к интеллектуальной ИТ быстрой разработки ИС и соответствующих проектных решений составила 9,22 человеко-месяца. Для ИС «Виртуальная доска объявлений» экономия трудозатрат за счет использования данной ИТ для повторного использования требований к ИАС «Реестр ЛУК» и соответствующих проектных решений составила 13,1 человеко-месяца. Показано, что интеллектуальная ИТ быстрой разработки ИС может осуществлять достаточно точную оценку трудозатрат на ИТ-проект создания ИС с учетом возможности повторного использования требований к ИС и соответствующих проектных решений на фазе инициации ИТ-проекта, когда сведения об архитектуре создаваемой ИС отсутствуют.

Предложенные в монографии концепции, модели и интеллектуальная ИТ быстрой разработки ИС позволяют решить важную научно-прикладную задачу эффективного повторного использования требований к ИС и соответствующих им проектных решений по информационному и программному обеспечениям на фазах инициации и планирования ИТ-проекта создания ИС, что позволит получить значительное сокращение затрат на реализацию подобных ИТ-проектов.

СВОД ОСНОВНЫХ ОПРЕДЕЛЕНИЙ

Архитектура [системы] – фундаментальные понятия и свойства системы в окружающей ее среде, воплощенные в ее элементах, отношениях, а также в принципах ее проектирования и развития (согласно стандарту ISO/IEC/IEEE 42010).

Архитектура ИТ-сервисов – фундаментальные понятия и свойства комплекса ИТ-сервисов и связывающих их интерфейсов, образующих ИС в окружающей её среде, воплощенные в элементах и отношениях этого комплекса, а также в принципах его проектирования и развития.

Архитектура ИТ-услуг – фундаментальные понятия и свойства комплекса ИТ-услуг, образующих ИС в окружающей её среде, воплощенные в элементах и отношениях этого комплекса, а также в принципах его проектирования и развития.

Архитектурный фреймворк ускоренной разработки информационной системы – архитектурный фреймворк, основанный на положениях сервисного подхода и предложенной концепции представления требований к ИС.

Главная цель деятельности ИС – формирование и отображение единого целостного информационного представления объекта или процесса в соответствии с поставленными перед системой целями.

Глобальная цель деятельности Поставщика ИТ-услуг – предоставление Потребителю ИТ-услуг такого набора ИТ-услуг, который наилучшим образом соответствует комплексу требований, определенных Потребителем и особенностями БП Потребителя, при ограничениях на стоимость, время выполнения, качество и содержание работ по предоставлению данного набора ИТ-услуг Потребителю.

Глобальная цель деятельности Потребителя ИТ-услуг – поиск и организация взаимодействия с таким Поставщиком ИТ-услуг, который предоставляет набор ИТ-услуг, наилучшим образом соответствующий комплексу требований, определенных Потребителем и особенностями БП Потребителя, при ограничениях на стоимость, время выполнения, качество и содержание работ по предоставлению данного набора ИТ-услуг выбранным Поставщиком.

Интеллектуальная информационная технология ускоренной разработки информационных систем – это совокупность унифицированных в

рамках архитектурного фреймворка ускоренной разработки ИС и основанных на знаниях методов, приемов и способов применения комплекса средств автоматизации или его отдельных элементов, используемую для выполнения одной или нескольких работ в рамках процессов создания ИС, непосредственно работающих с требованиями к данной системе.

Информационная система (ИС) – система, состоящая из персонала и комплекса средств автоматизации, и направленная на достижение главной цели деятельности ИС.

Информационная технология (ИТ) – совокупность унифицированных методов, приемов и способов применения комплекса средств автоматизации или его отдельных элементов, используемая для выполнения одной или нескольких функций ИС.

ИТ-сервис – совокупность различных средств комплекса средств автоматизации, реализующих законченную операцию предоставления или обработки данных, переводя их из одного целостного состояния в другое, используя при этом стандартные платформо-независимые интерфейсы.

ИТ-услуга (определение № 1) – это самостоятельная функциональная задача ИС, использование которой для выполнения отдельной работы процесса предприятия/организации или для управления этой работой экономически и технически целесообразно.

ИТ-услуга (определение № 2) – взаимосвязанная совокупность ИТ-сервисов, которая предоставляется для выполнения отдельной работы процесса предприятия/организации или для управления этой работой.

Комплекс средств автоматизации – вычислительное и коммуникационное оборудование, программное обеспечение, лингвистические средства и информационные ресурсы, нормативные и распорядительные документы и прочие средства, обеспечивающие достижение главной цели деятельности ИС.

Концепция представления требований к ИС определена как набор следующих положений:

а) отказ от рассмотрения только множества сформулированных требований к ИС и изначальное представление требований к ИС как элементов универсума, включающего в себя как известные, так и неизвестные

Поставщику, Потребителю или им обоим требования к ИС, а также методы формирования этих требований;

б) изначальное многообразие представлений требований к ИС в виде данных, информации и знаний;

в) процессный подход к описанию требований, определяющий минимальную процессную атрибутивную модель требования к ИС;

г) подход к управлению требованиями к ИС, основанный на изложенном в подразделе 1.6 основном принципе управления требованиями.

Основной принцип управления требованиями – постепенное преобразование множества начальных значений атрибутов, описывающих требование в множество желаемых значений те же атрибутов. Под желаемым значением следует понимать значение, которое приобретает атрибут при описании реализованного требования, проверенного соответствующими тестами.

Паттерн проектирования требований к ИС – результат выделения и повторного использования с прикладными целями Поставщиком ИТ-услуг следующих видов знаний:

а) об условии или возможности, которые необходимы Потребителю ИТ-услуг для решения стоящей перед ним проблемы или достижения поставленной перед ним цели;

б) об условии или возможности, которой должна обладать ИС или компонент ИС (ИТ-услуга, ИТ-сервис) с точки зрения Поставщика или Потребителя ИТ-услуг, соответствующих договору, стандарту, спецификации или другому официальному документу;

в) о документированном (с использованием естественного или формального языка) представлении условия или возможности подобно описанным в первых двух определениях.

Поставщик ИТ-услуг – организация и/или подразделение, которое основной целью своей деятельности считает эффективное и качественное выполнение работ по созданию, внедрению, эксплуатации и модернизации ИС, отдельных ИТ-услуг и реализующих эти услуги ИТ-сервисов.

Потребитель ИТ-услуг и реализующих эти услуги ИТ-сервисов – организация и/или подразделение, которое нуждается в этих услугах и сервисах и использует их в своих процессах для достижения целей, поставленных перед этой организацией и/или подразделением.

Процесс – совокупность взаимосвязанных и взаимодействующих видов деятельности, преобразующих входы (материальные, информационные потоки и т.п.) в выходы, представляющие ценность для потребителя (согласно стандарту ISO 9000:2005).

Публикация требования к ИС – это:

а) описание условия или возможности, необходимых Потребителю ИТ-услуг для решения проблемы или достижения цели, выполненное одним из допустимых в рамках методологии разработки ИС способов;

б) описание условия или возможности, которой должна обладать ИС или компонент ИС (ИТ-услуга, ИТ-сервис) с точки зрения Поставщика или Потребителя ИТ-услуг, соответствующих договору, стандарту, спецификации или другому официальному документу, выполненное одним из допустимых в рамках методологии разработки ИС способов.

Требование – это:

- условие или возможность, необходимые пользователю для решения проблемы или достижения цели;
- условие или возможность, которой должна обладать система или компонент системы, соответствующие договору, стандарту, спецификации или другому официальному документу;
- документированное представление условия или возможности подобно описанному в первых двух определениях (согласно стандарту IEEE 610.12-1990).

Требование к ИС – это:

а) условие или возможность, необходимые Потребителю ИТ-услуг для решения проблемы или достижения цели;

б) условие или возможность, которой должна обладать ИС или компонент ИС (ИТ-услуга, ИТ-сервис) с точки зрения Поставщика или Потребителя ИТ-услуг, соответствующие договору, стандарту, спецификации или другому официальному документу;

в) документированное представление условия или возможности подобно описанному в первых двух определениях.

Универсум – это совокупность данных, информации и знаний об исследуемой системе, объекте или процессе, как известных, так и неизвестных исследователю, в распоряжении которого имеется конечное множество методов добычи и обработки этих данных, информации и знаний.

Функциональная задача – задача, решение которой позволяет реализовать соответствующую функцию ИС.

Функция ИС – совокупность операций ИС, выполняемых над данными и направленных на достижение поставленной перед ИС конкретной цели или подцели.

Приложение А

ОПИСАНИЕ ПРОЦЕССОВ МАКРОПРОЕКТИРОВАНИЯ ИНФОРМАЦИОННОЙ СИСТЕМЫ

В данном приложении в табл. А.1 – А.4 приведены описания процессов, непосредственно работающих с требованиями, в соответствии с положениями методологии SSADM V4.2, а также международных стандартов ISO/IEC 15288:2002 и ISO/IEC 12207:2008.

Необходимо отметить, что для методологии SSADM понятие «процесс» не является общепринятым. Данная методология использует понятие «стадия». Однако в современных стандартах, определяющих методологию SSADM, описания каждой стадии соответствуют принятой и действующей в настоящее время модели описания процесса.

В качестве модели описания процесса использована концептуальная модель, задекларированная в ISO/IEC 24774:2007.

В табл. А.5 – А.6 приведены описания процессов проектирования архитектуры системы в соответствии с положениями стандартов также международных стандартов ISO.

Таблица А.1 – Описание процессов, непосредственной работающих с требованиями, в группе технических процессов стандарта ISO/IEC 15288:2002

Наименование процесса	Цели процесса	Выходы процесса	Виды деятельности процесса
1	2	3	4
Процесс определения требований правообладателей.	Выявление требований к системе, выполнение которых может обеспечить функциональные возможности, необходимые пользователям системы и иным заинтересованным лицам в заданной эксплуатационной среде.	а) задаются требуемые характеристики и условия использования функциональных возможностей системы; б) определяются ограничения для системных решений; в) достигается возможность текущего отслеживания связей между требованиями правообладателей и самими правообладателями и их потребностями; г) описывается основа для определения системных требований;	- идентификация отдельных правообладателей или классов правообладателей, имеющих законный интерес к системе в течение ее жизненного цикла; - выявление требований правообладателей; - определение ограничений системных решений, которые являются неизбежным следствием существующих соглашений, управленческих или технических решений;

Продолжение табл. А.1

1	2	3	4
		д) определяется основа для валидации соответствия функциональных возможностей системы; е) формируется основа для ведения переговоров и заключения соглашения о поставке продукции или услуг.	- определение представительного набора последовательных действий для идентификации всех требуемых функциональных возможностей, которые отвечают предполагаемым сценариям и средам функционирования и сопровождения; - определение взаимодействия между пользователями и системой; - установление и спецификация экологических, медицинских требований, требований безопасности и других требований правообладателей, имеющие отношение к критическим показателям; - анализ полной совокупности выявленных требований; - разрешение проблем, возникших в связи с определением требований; - доведение результатов анализа требований до сведения соответствующих правообладателей для гарантии того, что их потребности и ожидания были правильно поняты и выражены; - установление совместно с правообладателями корректности выражения их требований; - документировать требования правообладателей в форме, приемлемой для управления требованиями в течение жизненного цикла и за его пределами;

Продолжение табл. А.1

1	2	3	4
			- поддерживать взаимное соответствие между требованиями правообладателей и потребностями заинтересованных лиц.
Процесс анализа требований.	Преобразование требований правообладателя, выраженных в виде его представлений о желаемых функциональных возможностях, в техническое видение требуемого продукта, способного предоставить такие функциональные возможности.	а) устанавливаются требуемые характеристики, свойства, функциональные и эксплуатационные требования к техническим решениям; б) устанавливаются ограничения, влияющие на архитектурное проектирование системы, а также на средства по его реализации; в) достигается целостность и прослеживаемость системных требований к требованиям правообладателей; г) определяется основа для верификации системных требований.	- определение функциональных границ системы в терминах ее поведения и свойств, которые должны быть обеспечены; - определение каждой функции, которую система должна выполнять, насколько хорошо система, включая операторов, должна выполнять эту функцию, а также условий, при которых система способна выполнять данную функцию и при которых система начинает и прекращает ее выполнение; - определение необходимых ограничений по изготовлению системы и ее элементов, которые обусловлены требованиями правообладателей или неизбежными ограничениями, связанными с принятием решений; - определение технических показателей и показателей качества при использовании, позволяющих оценивать технические достижения; - установление системных требований и функций, в соответствии с которыми определяются риски и критические параметры системы, связанные с такими свойствами, как здоровье, безопасность,

Продолжение табл. А.1

1	2	3	4
			защищенность, безотказность, готовность, а также со свойствами обеспечивающих систем; - анализ целостности системных требований для обеспечения уверенности в том, что каждое требование, пары требований или наборы требований обладают системной целостностью; - демонстрация связи между системными требованиями и требованиями правообладателей; - на протяжении всего жизненного цикла ведение учета совокупности системных требований вместе с их обоснованиями, связанными решениями и допущениями.

Таблица А.2 – Описание процессов, непосредственно работающих с требованиями к программным системам, в группе технических процессов стандарта ISO/IEC 12207:2008

Наименование процесса	Цели процесса	Выходы процесса	Виды деятельности процесса
1	2	3	4
Процесс определения требований правообладателей.	Выявление требований к системе, выполнение которых может обеспечивать предоставление услуг, необходимых пользователям и другим правообладателям в заданной среде применения.	а) задаются требуемые характеристики и условия использования услуг; б) определяются ограничения для системных решений; в) достигается возможность прослеживания от требований правообладателей к правообладателям и их потребностям; г) описывается основа для определения системных требований;	- идентификация правообладателей; - идентификация требований; - оценка требований; - согласование требований; - регистрация требований.

Продолжение табл. А.2

1	2	3	4
		д) определяется основа для валидации соответствия услуг; е) формируется основа для ведения переговоров и заключения соглашений о поставке услуги или продукции.	
Процесс анализа системных требований.	Преобразование определенных требований правообладателей в совокупность необходимых системных технических требований, которыми будут руководствоваться в проекте системы.	а) устанавливается определенная совокупность системных функциональных и нефункциональных требований, описывающих проблему, подлежащую решению; б) выполняются соответствующие технические приемы оптимизации предпочитаемого проектного решения; в) системные требования анализируются на корректность и тестируемость; г) осмысливается воздействие системных требований на среду применения; д) требования расставляются по приоритетам, утверждаются и обновляются; е) устанавливается согласованность и прослеживаемость между системными требованиями и базовой линией требований заказчика; ж) оцениваются изменения базовой линии по стоимости, графикам работ и воздействию технических решений; и) системные требования доводятся до сведения всех участвующих сторон и включаются в базовую линию.	- спецификация требований; - оценивание требований.

Таблица А.3 – Описание процессов, непосредственной работающих с требованиями к программным средствам, в группе процессов жизненного цикла программных средств стандарта ISO/IEC 12207:2008

Наименование процесса	Цели процесса	Выходы процесса	Виды деятельности процесса
1	2	3	4
Процесс анализа требований к программным средствам.	Установление требований к программным элементам системы.	а) определяются требования к программным элементам системы и их интерфейсам; б) требования к программным средствам анализируются на корректность и тестируемость; в) осознается воздействие требований к программным средствам на среду функционирования; г) устанавливается совместимость и прослеживаемость между требованиями к программным средствам и требованиями к системе; д) определяются приоритеты реализации требований к программным средствам; е) требования к программным средствам принимаются и обновляются по мере необходимости; ж) оцениваются изменения в требованиях к программным средствам по стоимости, графикам работ и техническим воздействиям; и) требования к программным средствам воплощаются в виде базовых линий и доводятся до сведения заинтересованных сторон.	- анализ требований к программным средствам, в том числе: а) установить и документально оформить следующие требования к программным средствам (включая спецификации характеристик качества); б) оценить требования к программным средствам, учитывая заданные стандартом критерии, и документально оформить результаты оценивания; в) проводить ревизии.

Таблица А.4 – Описание стадий, непосредственно работающих с требованиями, согласно методологии SSADM

Наименование стадии	Цели стадии	Выходы стадии	Виды деятельности стадии
1	2	3	4
Стадия 0 «Осуществимость» (Feasibility).	- определение степени удовлетворения существую-	- описание существующей системы; - модель существующих	- определение проблем; - выбор варианта реализуемости ИС.

Продолжение табл. А.4

1	2	3	4
	щей ИС выдвинутым бизнес-требованиям организации; - формирование экономического обоснования предлагаемой ИС; - обеспечение для комитета управления проектом создания ИС возможности выбора из ряда вариантов реализуемости ИС; - определение необходимости выделения ресурсов для более детального изучения предметной области, существующей и разрабатываемой ИС.	физических потоков данных; - логическая модель данных текущей ситуации; - каталог требований; - требования к аудиту, контролю и безопасности; - список ограничений; - каталог пользователей; - выбранный вариант реализуемости ИС.	
Стадия 1 «Исследование текущей ситуации» (Investigation of Current Environment).	- понимание текущей ситуации и разработка описаний текущих услуг; - выявление проблем, связанных с текущей ситуацией, которые должны быть решены в новой системе; - выявление дополнительных услуг, предоставляемых новой системой; - обеспечение логического представления текущих услуг; - установление ролей, особенно для пользователей в проекте.	- модель бизнес-деятельности; - описание существующей системы; - модель существующих физических потоков данных; - логическая модель данных текущей ситуации; - каталог пользователей; - каталог требований; - требования к аудиту, контролю и безопасности; - список ограничений.	- разработка модели бизнес-деятельности; - изучение и определение требований; - исследование текущих способов обработки (текущего процессинга); - исследование существующих описаний данных; - формирование логического представления существующих услуг.

Продолжение табл. А.4

Стадия 2 «Выбор бизнес-системы» (Business System Option).	- предложение ряда вариантов требуемой системы в соответствии с приоритетами, установленными в каталоге требований; - оказание помощи комитету управления проектом в выборе лучшего решения из предложенных вариантов системы.	- выбранный вариант бизнес-системы.	- определение вариантов бизнес-системы; - выбор вариантов бизнес-системы.
Стадия 3 «Определение требований» (Definition of Requirements).	Детальная спецификация требований к данным и к процессам их обработки, выдвинутых к выбранному варианту бизнес-системы.	- описание требуемой системы; - модель потоков данных требуемой системы; - логическая модель данных требуемой системы; - определения функций; - модель практик работы (документирует понимание пользователем своей работы на предприятии).	- определение процессов обработки данных требуемой системы; - разработка модели необходимых данных; - установление функций системы; - разработка спецификаций работы пользователей; - совершенствование модели требуемых данных; - разработка специфицирующих прототипов; - разработка спецификаций процессов обработки данных; - подтверждение целей системы.

Таблица А.5 – Описание процесса проектирования архитектуры из группы технических процессов стандарта ISO/IEC 15288:2002

Наименование процесса	Цели процесса	Выходы процесса	Виды деятельности процесса
1	2	3	4
Процесс проектирования архитектуры.	Синтез решения, которое бы удовлетворяло системным требованиям.	а) устанавливается порядок, в соответствии с которым выполняется проектирование архитектуры; б) задается реализуемый набор описаний системных элементов, которые удовлетворяют требованиям,	- определять приемлемые проекты логической архитектуры; - выполнять декомпозицию функций системы, определенных в процессе анализа требований, и поставить им в соответ-

Продолжение табл. А.5

1	2	3	4
		предъявляемым к системе; в) включаются в решение по проектированию архитектуры требования к интерфейсу; г) устанавливается связь между проектированием архитектуры и системными требованиями; д) определяется основа для верификации системных элементов; е) устанавливается основа комплексирования системных элементов.	ствие элементы архитектуры системы, сформировать производные требования, необходимые для такого сопоставления; - анализировать итоговый проект архитектуры с целью установления проектных критериев для каждого элемента; - определять, какие системные требования должны выполняться операторами; - определять, доступны ли в готовом виде те элементы технического и программного обеспечения, которые удовлетворяют проектным и интерфейсным критериям; - оценивать альтернативные проектные решения, моделируя их с той степенью детализации, которая позволяет сравнивать спецификации, выраженные в системных требованиях, с эксплуатационными характеристиками, стоимостными и временными показателями и рисками, выраженными в требованиях правообладателей; - определять и документировать области взаимодействия между системными элементами и области взаимодействий на границе системы с внешними системами; - задавать выбранные физические проектные решения в соответствии с порядком проектирования архитектуры в терминах проектных функций,

Продолжение табл. А.5

1	2	3	4
			характеристик эксплуатации, поведения, интерфейсов и неизбежных ограничений при реализации проекта; - вести документальный учет информации по проектированию архитектуры; - поддерживать взаимосвязь и взаимозависимость между архитектурой и системными требованиями.

Таблица А.6 – Описание процессов проектирования архитектуры системы и проектирования архитектуры программных средств стандарта ISO/IEC 12207:2008

Наименование процесса	Цели процесса	Выходы процесса	Виды деятельности процесса
1	2	3	4
Процесс проектирования архитектуры системы.	Определение того, как системные требования следует распределить относительно элементов системы.	а) определяется архитектурный проект системы, в соответствии с которым выполняется идентификация элементов системы и удовлетворяются заданные требования; б) устанавливаются функциональные и нефункциональные системные требования; в) требования распределяются по элементам системы; г) определяются внутренние и внешние интерфейсы каждого системного элемента; д) выполняется верификация между системными требованиями и архитектурой системы; е) требования, распределенные по системным элементам и их интерфейсам, ста-	- создание архитектуры; - оценивание архитектуры.

Продолжение табл. А.6

1	2	3	4
		новятся прослеживаемыми к базовой линии требований заказчика; ж) поддерживается согласованность и прослеживаемость между системными требованиями и архитектурным проектом системы; и) системные требования, конструкция, архитектурный проект системы и их взаимосвязи отражаются в базовой линии и сообщаются всем участвующим сторонам; к) в системный проект включается человеческий фактор, эргономические знания, технические приемы, методы и средства; л) определяются и выполняются действия по проектированию, ориентированные на человека.	
Процесс проектирования архитектуры программных средств.	Обеспечение проекта для программных средств, которые реализуются и могут быть верифицированы относительно требований.	а) разрабатывается проект архитектуры программных средств и устанавливается базовая линия, описывающая программные составные части, которые будут реализовывать требования к программным средствам; б) определяются внутренние и внешние интерфейсы каждой программной составной части; в) устанавливаются согласованность и прослеживаемость между требованиями к программным средствам и программным проектом.	- проектирование архитектуры программных средств, которое заключается в следующих действиях: а) исполнитель должен преобразовать требования к программным составным частям в архитектуру, которая описывает верхний уровень его структуры и идентифицирует программные компоненты. Необходимо гарантировать, что все требования к программным составным частям распределяются по программным компонентам и в дальнейшем уточняются для облегчения детального проектирования. Архитектуру программной составной части необходимо документировать;

Продолжение табл. А.6

1	2	3	4
			б) исполнитель должен разработать и документально оформить проект верхнего уровня для внешних интерфейсов программной составной части и интерфейсов между ней и программными компонентами; в) исполнитель должен разработать и документально оформить проект верхнего уровня для базы данных; г) исполнитель должен разработать и документально оформить предварительные версии пользовательской документации; д) исполнитель должен определить и документировать требования к предварительному тестированию и график работ по комплексированию программных средств; е) исполнитель должен оценить архитектуру программной составной части, проекты по интерфейсам и базе данных, учитывая следующие критерии: прослеживаемость к требованиям программной составной части; внешняя согласованность с требованиями программной составной части; внутренняя согласованность между программными компонентами; приспособленность методов проектирования и используемых стандартов; осуществимость детального проектирования; осуществимость функционирования и сопровождения. Результаты оценок следует оформлять документально; ж) исполнитель должен проводить ревизии.

Приложение Б

ПРИМЕР XSL-ТРАНСФОРМАЦИИ XML-ДОКУМЕНТА В SQL-ЗАПРОСЫ

Рисунок Б.1 – Фрагмент исходного кода XML-документа, описывающего UML-диаграмму классов, сформированную в CASE-средстве ModelMaker for Embarcadero Delphi

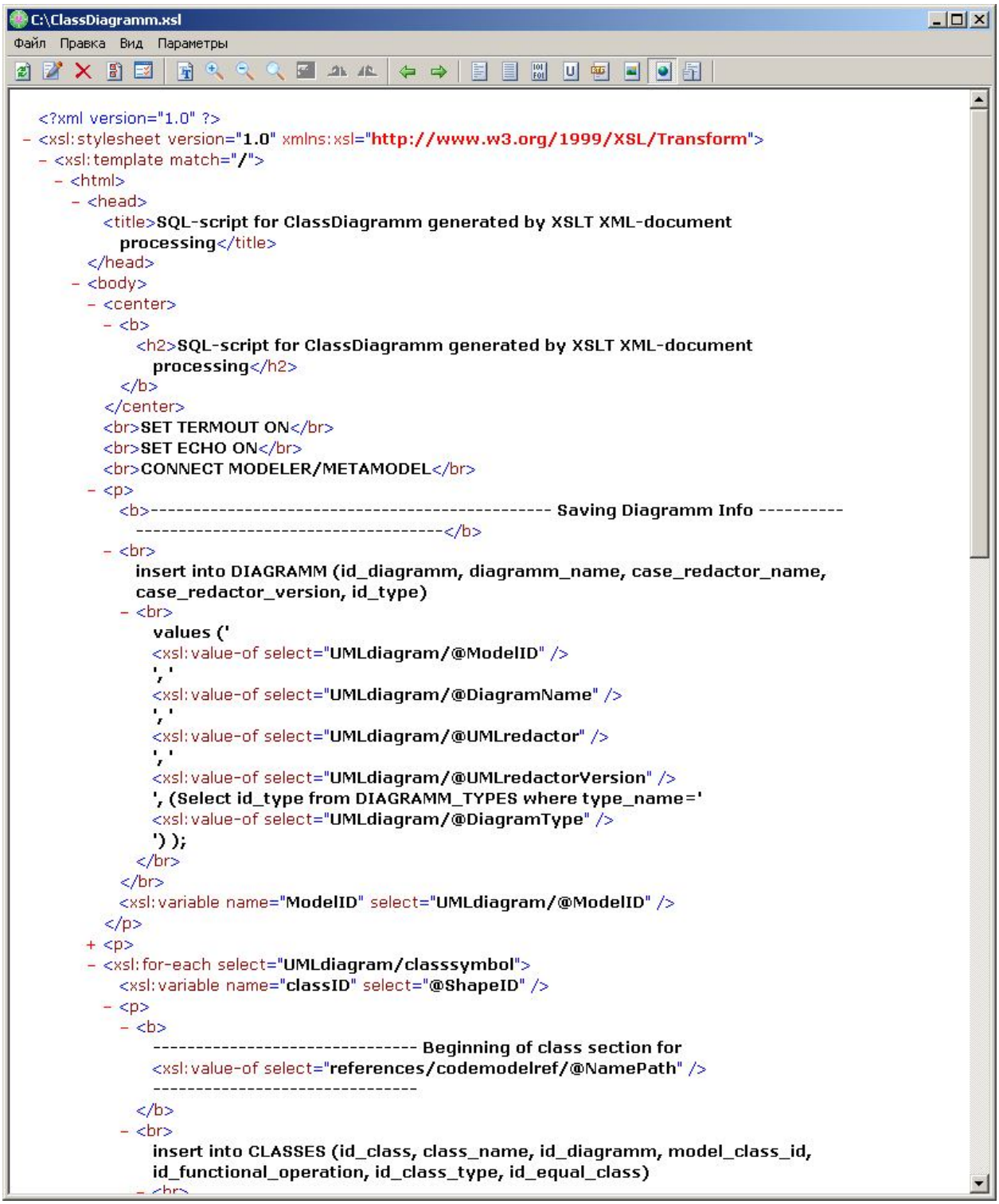

Рисунок Б.2 – Фрагмент исходного кода XSLT-документа, описывающего трансформацию XML-документа в набор INSERT-команд заполнения базы данными об элементах диаграммы классов

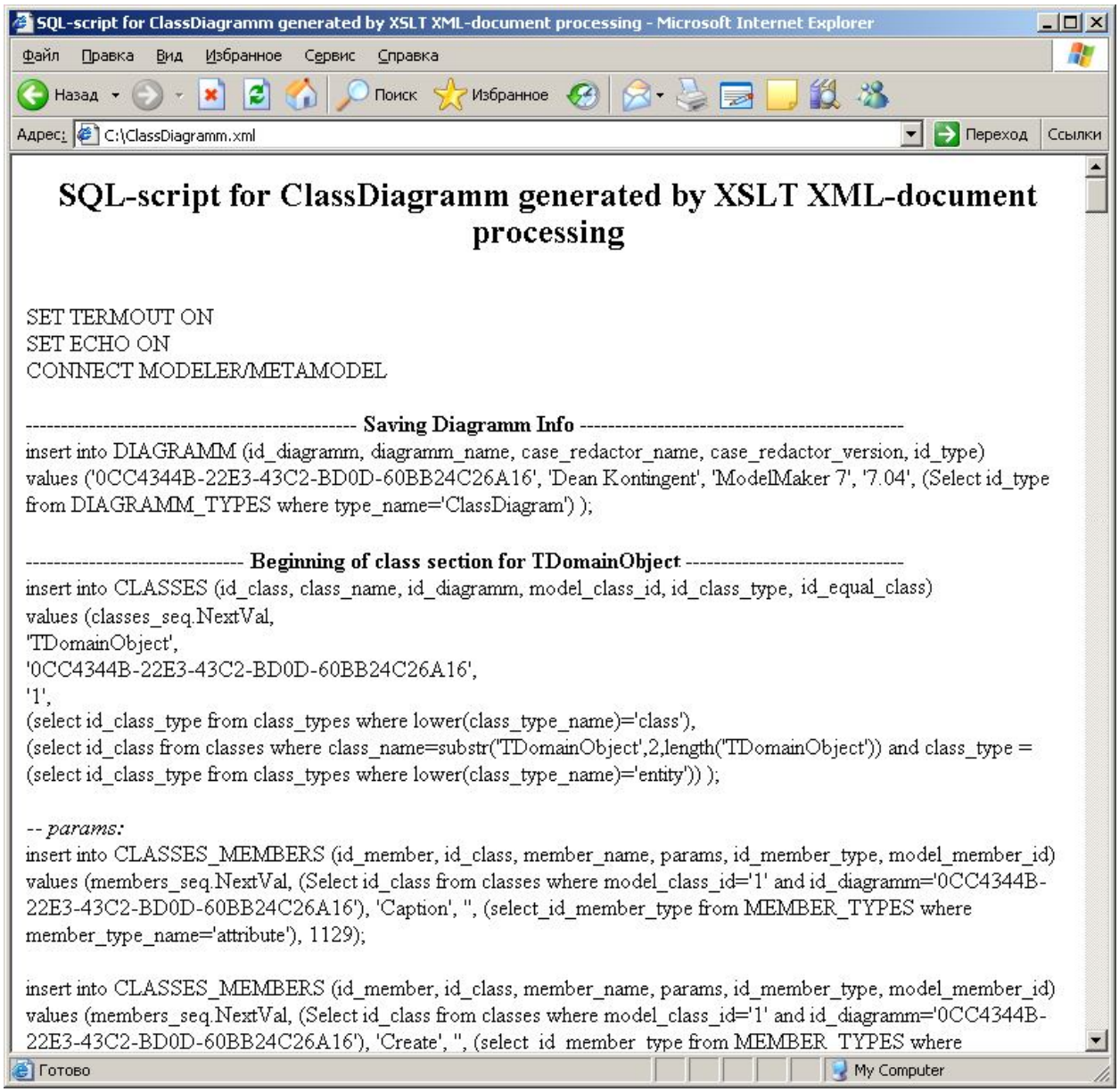

Рисунок Б.3 – Результат трансформации XML-документа в набор INSERT-команд заполнения базы данными об элементах диаграммы классов

Приложение B

ПРИМЕР СОГЛАШЕНИЯ ОБ ИМЕНОВАНИИ ОБЪЕКТОВ БАЗЫ ДАННЫХ

==========
========== ВСЕ ОБЪЕКТЫ БД ==========
==========

0) Правило английского языка
Все объекты БД, включая названия, поля таблиц, триггеры, представления, процедуры и т.д. именуются, с применением правильного английского языка. Калька с русского (русские слова, записанные английскими буквами) не используется.

0.1) Правило разделения слов
Если имя объекта БД или его элемента (таблицы, поля, ограничения, процедуры, функции, триггера, пакета, последовательности и т.д.) состоит из нескольких слов, слова всегда разделяются символом подчеркивания.

Например:
Таблица *AGE_CATAGORY*,
Поле *AGE_CATEGORY_NAME*.

0.2)Для проверки выполнения большинства правил применительно к уже существующим объектам БД используются объекты схемы *APP_CORE*: представления серии *V_UTIL_...*, процедуры серии *P_UTIL_...*, функции серии *F_UTIL_...* (имя объекта указывается в скобках рядом с названием правила).
Автоматизированную проверку всех описанных далее правил обеспечивает вызов процедуры *P_UTIL_CHK_ALL*, которая при нарушении правил выводит в лог название и содержимое представления, описывающего аномалии не выполнившегося правила.

==========
========== ТАБЛИЦЫ ==========
==========
1)Имена таблиц должны совпадать с именами бизнес-классов в ПО.

1.0)Правило единственного числа (проверка - *V_UTIL_CHK_TABLE_NAMES*).
Имена таблиц всегда пишутся в единственном числе.

Например:
PERSON.

Правило зарезервированных слов (проверка - *V_UTIL_CHK_TABLE_NAMES* на основе *F_UTIL_GET_TABLE_NAME*).
Исключением из правила единственного числа являются имена таблиц, совпадающие с зарезервированными словами (*FILE, SESSION, USER*) и т.д.
Такие таблицы создаются с именем, заданным во множественном числе.

Например:
FILE->FILES.

При дальнейшей работе с такими таблицами (при создании столбцов, ключей) считается, что их название задано в ЕДИНСТВЕННОМ ЧИСЛЕ, и правила именования элементов данной таблицы вычисляются относительно названия таблицы, взятого в единственном числе.

Например:
Таблица *FILES*, первичный ключ: *PK_ID_FILE*.

1.1) Правило энумераторов (проверка - *V_UTIL_CHK_TABLE_NAMES* на основе *F_UTIL_GET_TABLE_NAME*)
Имена таблиц, соответствующих энумераторам в ПО, должны быть заданы в формате "*ИМЯ_ЭНУМЕРАТОРА_DICT*".
ИМЯ_ЭНУМЕРАТОРА_DICT = ИМЯ_ТАБЛИЦЫ.

Например:
Класс-энумератор *Gender* => таблица *GENDER_DICT*,
Класс-энумератор *PopulationGroup* => таблица *POPULATION_GROUP_DICT*.

При этом суффикс *_DICT* является только флагом того, что таблица соответствует энумератору.
В качестве имени таблицы при именовании ограничений и полей используется название таблицы без суффикса.

==========
========== ПОЛЯ ТАБЛИЦ ==========
==========2)

Правило повторяющихся столбцов (проверка - *V_UTIL_CHK_COLUMN_NAMES*).
Поля, имена которых часто встречаются (во многих таблицах, например, «название»), следует задавать в формате
"*ИМЯ_ТАБЛИЦЫ_ИМЯ_ПОЛЯ*".

Например:
AGE_CATEGORY_NAME,
MEETING_NAME.
На поля с суффиксом "*_DATE*" данное правило не распространяется.
Оправданные исключения фиксируются в *V_UTIL_CHK_COLUMN_NAMES*.

2.1) Правило столбцов-идентификаторов (проверка - *V_UTIL_CHK_COLUMN_NAMES*).
Поля, вводимые для создания искусственных первичных ключей, именуются следующим образом:
"*ID_ИМЯ_ТАБЛИЦЫ*".

Например:
ID_PERSON,
ID_TYPE.

Подправило идентификаторов иерархий (проверка - *V_UTIL_CHK_COLUMN_NAMES*)
Поля, являющиеся внешними ключами, ссылающимися на эту же таблицу (т.е. использующиеся для создания реляционных иерархических связей), именуются следующим образом:
"*ID_PARENT_ИМЯ_ТАБЛИЦЫ*"

Подправило внешних идентификаторов (проверка - *V_UTIL_CHK_COLUMN_NAMES*)
Поля, являющиеся внешними ключами, должны совпадать по имени с полями, на которые они ссылаются (исключением является ситуация, когда в таблице есть несколько внешних ключей, ссылающихся на одну и ту же таблицу, или переименования поля оправдано).

Например:
Таблица *PLAYING_REGION_DICT* содержит поле *ID_PLAYING_REGION*, являющееся первичным ключом, а также поле *ID_EQUAL_REGION, которое* ссылается на географический регион, являющийся эквивалентом игрового ("*APP_GEOGRAPHY*"."*REGION*"."*ID_REGION*").
Имеет место наличие двух столбцов с идентификатором региона. В этом случае в имя столбца, как правило, добавляется уточняющее слово.

Если поля являются первичными и внешними ключами одновременно, то правило именования внешних ключей является приоритетным.
См. таблицы *CONTRACT && CONTRACT_LUK схемы данных ИАС ВОО «ЛУК»* - в обеих таблицах первичным ключом являются поля *ID_CONTRACT*.

==========
========== ОГРАНИЧЕНИЯ ==========
==========3)

Ограничения (первичные, внешние, уникальные ключи, пользовательские ограничения, за исключением not null) должны иметь понятные имена с обязательными префиксами (задаются вместо названий типа *SYS_C0020459*, автоматически генерируемых СУБД):

*PK_* для первичного ключа;
*FK_* для внешнего ключа;
*UK_* для уникального ключа;
*CHK_* для пользовательского ограничения.

В качестве имени ограничения следует указывать имя в формате
"*префикс + имя_поля_участвующего_в_ограничении*".

Например: *FK_ID_PERSON*.

3.1) Составные ограничения (например, составные ключи, когда в ограничении участвует не одно поле) создаются с именем в формате "*префикс + имя_таблицы*".

Например:
UK_AGE_CATEGORY_PAYMENT,
PK_TEAM_REQUEST.

3.2) Если имя ограничения уже существует в другой таблице той же схемы данных, то используется формат "*префикс + имя_таблицы + имя_поля*".

Например:
FK_TEAM_MEMBERSHIP_ID_PERSON.

3.3) Для внешних ключей, ссылающихся на слабые сущности, в конце следует добавлять суффикс слабой сущности (чтобы было понятно на какую из таблиц с одинаковыми первичными ключами создается ссылка).

Например:
FK_MEMBERSHIP_ID_PERSON_LUK (суффикс - *_LUK*).

3.4) Для иерархических связей (внешних ключей таблицы, ссылающихся на неё же) используется формат:
"*FK_ИМЯ_ПОЛЯ*"

Например, для поля *ID_PARENT_TYPE* таблицы *TYPE*:
FK_ID_PARENT_TYPE.

3.5)Когда первичный ключ создается для слабой сущности, то вычисленное по предыдущим правилам имя ограничения уже может быть занято. В таком случае используется формат:
"*префикс + имя_таблицы*" (как в случае с составными ограничениями).
См. *CONTRACT && CONTRACT_LUK*. В таблице *CONTRACT* есть первичный ключ *PK_ID_CONTRACT*. Для таблицы *CONTRACT_LUK* первичный ключ, состоящий из того же поля *ID_CONTRACT* будет иметь имя *PK_CONTRACT_LUK*.

3.6)Если название ограничения превышает допустимую длину - вначале сокращается часть, соответствующая именам столбцов, только потом сокращается часть, соответствующую имени таблицы.
Например:
FK_PERSON_REQUEST_ID_TOURNAMENT_REQUEST => FK_*PERSON_REQUEST_ID_TOURN_REQ*.

3.7)Если пользовательское ограничение обеспечивает обязательное совместное заполнение пары NULL-полей одной таблицы, оно должно иметь имя вида:
CHK_BOTH_NULL_имя

Например:
CHK_BOTH_NULL_END для таблицы *APP_MEMBERSHIP.MEMBERSHIP* (поля *END_DATE* и *ID_REASON_END* с датой и причиной исключения члена организации).

3.8)Если пользовательское ограничение обеспечивает обязательное заполнение ТОЛЬКО ОДНОГО ИЗ пары NULL-полей одной таблицы, оно должно иметь имя вида:
CHK_ONE_NULL_имя

Например:
CHK_ONE_NULL_SESSION_MEETING для таблицы *APP_LUK_CONTRACT* в которой поля *ID_SESSION* и *ID_MEETING* не могут быть заполнены одновременно, но могут одновременно быть пустыми.

==========
========== ИНДЕКСЫ ==========
==========
4) Индексы, соответствующие ограничениям, создаются с именами в формате "*X_ИМЯ_ОГРАНИЧЕНИЯ*".

Например:
X_PK_AGE_CATEGORY_PAYMENT.

4.1) Индексы, используемые для повышения скорости выполнения AJAX-запросов, создаются с именами в формате "*X_AJAX_ИМЯ_ТАБЛИЦЫ*"

Например:
X_AJAX_SETTLEMENT.

4.2) Индексы для внешних ключей создаются с именами в формате "*X_FK_имя_таблицы_имя_столбца*"
Например:*X_FK_VOTE_ID_PUBLICATION*.

==========
========== ХРАНИМЫЕ ПРОЦЕДУРЫ ==========
==========
5) Все хранимые процедуры для добавления\изменения\удаления записей в таблицы БД называются в формате "*P_ИМЯ_ТАБЛИЦЫ*"

Например:
P_CHAMPIONSHIP_STRUCTURE

5.1) Остальные хранимые процедуры называются в формате "P__ИМЯ_ХРАНИМОЙ_ПРОЦЕДУРЫ" (два подчеркивания в начале).

5.2) В структуре хранимых процедур первый входной параметр всегда *ipAction*, второй - столбец с ID (если используется несоставной первичный ключ).

Например:
```
PROCEDURE p_user
( ipAction in number,
  ipID_ USER in users.ID_USER%TYPE,
  ...
)
```

==========
========== ФУНКЦИИ ==========
==========6)
Все хранимые функции называются в формате "*F_ИМЯ_ФУНКЦИИ*".
6.1) Функции, используемые функциональными индексами, называются в формате "*F_GET_ИМЯ_ФУНКЦИИ*".

==========
========== ПРЕДСТАВЛЕНИЯ ==========
==========
7) Все представления называются в формате *"V_ПРЕФИКС_ИМЯ_ТАБЛИЦЫ_СУФФИКС"*.
Префикс в названии может быть только один.
Суффиксов одновременно может быть много. Используется следующий порядок их задания:
1) *_FULL*
2) *_H*

Например:
ПРЕФИКС_ИМЯ_ТАБЛИЦЫ_FULL_H.

7.1) Все основные представления, используемые DAO для загрузки данных в объекты ПО (выбирают данные только из **одной** таблицы), называются с использованием префикса *"V_T_"*.

Например:
V_T_FAMILY_STATUS.

7.2) Все основные представления, используемые DAO для загрузки энумераторов (выбирают данные из **одной** таблицы), называются с использованием префикса *"V_DICT_"*.

Например:
V_DICT_GENDER.

7.3) Все дополнительные представления, используемые для получения специфических сведений обо всех записях **нескольких** таблиц, называются с использованием префикса *"ALL_"* в формате *"V_ALL_СМЫСЛОВОЕ_НАЗВАНИЕ"*.

Например:
V_ALL_ABOUT_MEMBERS.

7.4) Все представления, которые не используются в ПО и нужны только для администрирования БД, называются с использованием префикса *" UTIL_"*.

Например:
V_UTIL_PAGE_COMPOSITION.

7.5) Представления, содержащие внешние соединения, называются с использованием суффикса *"_FULL"*.

Например:
V_ALL_PRIVILEGE_GRANT_FULL.

7.6) Представления, содержащие иерархические запросы, называются с использованием суффикса *"_H"*.

Например:
V_ALL_PRIVILEGE_GRANT_FULL_H

==========
========== ПОСЛЕДОВАТЕЛЬНОСТИ ==========
==========8)
Все последовательности, используемые хранимыми процедурами для добавления новых записей в таблицы БД, называются в формате "*SEQ_имя_ключевого_столбца*".
Например: *SEQ_ID_COUNTRY*.

Одна последовательность используется для заполнения только одной таблицы (слабые сущности используют значения первичных ключей сильных сущностей).

==========
========== ТРИГГЕРЫ ==========
==========9)
Все триггеры именуются в формате "*T_имя_триггера*".

9.1) Системные триггеры именуются в формате "*T_SYS_имя_триггера*".

Например:
T_SYS_LOGON.

9.2) Триггеры, обеспечивающие проверку корректности заполнения данных таблиц, создаются с именами вида
"*T_CHK_имя_таблицы_имя_поля*".

Например:
T_CHK_FAMILY_STATUS_PRIORITY.

9.3) Триггеры, обеспечивающих целостность данных в обобщённой (сильной) и уточняющих (слабых) таблицах, именуются следующим образом:

а) обеспечивающие сюрьективное отображение данных (триггеры, обеспечивающие связь 1:1).

Формат: *T_SYNC_ИМЯ_СИЛЬНОЙ_СУЩНОСТИ_* для сильной сущности, *T_SYNC_ИМЯ_СЛАБОЙ_СУЩНОСТИ* для слабой сущности.

Например:
*T_SYNC_CONTRACT_* для таблицы *CONTRACT*;
T_SYNC_CONTRACT_LUK для таблицы *CONTRACT_LUK*.

а2) Если таблица одновременно является и наследующей и наследуемой, то когда она выступает в роли слабой сущности используется имя формата *T_SYNC_ИМЯ_СУЩНОСТИ*, когда в роли сильной - *T_SYNC_ИМЯ_СУЩНОСТИ_*.

Например: иерархия фреймов *PERSON->PERSON_PHYSICAL->PERSON_PHYSICAL_LUK*
Триггеры:
*T_SYNC_PERSON_* для таблицы *PERSON*;
T_SYNC_PERSON_PHYSICAL для таблицы *PERSON_PHYSICAL* в роли слабой сущности;
*T_SYNC_PERSON_PHYSICAL_* для таблицы *PERSON_PHYSICAL* в роли сильной сущности;
T_SYNC_PERSON_PHYSICAL_LUK для таблицы *PERSON_PHYSICAL_LUK*.

б) обеспечивающие уникальность значений комбинаций атрибутов (аналог уникального ключа в одной таблице.

Формат: "*T_UNIQ_имя_зависимой_таблицы*".

Например:
T_UNIQ_TEAM_REQUEST.

в) обеспечивающие контроль внесения данных в таблицы (когда сильная сущность связана с несколькими слабыми, и данные в слабых сущностях не должны пересекаться между собой, связь 1:0..1), обеспечивающие связь одной сильной сущности с несколькими слабыми (разбиение множества на подмножества).

Формат: T_*SPLT*_ИМЯ_СИЛЬНОЙ_СУЩНОСТИ для сильной сущности, T_*SPLT*_ИМЯ_СЛАБОЙ_СУЩНОСТИ для каждой из слабых сущностей.

Например:
T_SPLT_PERSON ==> 1)*T_SPLT_PERSON_JURIDICAL* 2)*T_SPLT_PERSON_PHYSICAL*.

В случае многоуровневого деления именования триггеров осуществляется аналогично п.а2.
Ситуации, описанные в п.а2 и п.в, являются взаимоисключающими.

г) обеспечивающие заполнения необязательных полей сильных сущностей, если это требуется для зависящих от них слабых сущностей (аналог not null option).

Формат: "*T_NULL_ИМЯ_СУЩНОСТИ*".

Например:
T_NULL_PERSON_JURIDICAL.

Приложение Г

ПРИМЕРЫ ПРОЦЕДУР КОНТРОЛЯ СОБЛЮДЕНИЯ ПРАВИЛ ИМЕНОВАНИЯ ОБЪЕКТОВ БАЗЫ ДАННЫХ

На рис. Г.1 показан фрагмент процедуры-оболочки, обеспечивающей последовательное выполнение этапов контроля именования объектов базы данных и вывод найденных несоответствий.

```
PROCEDURE p_util_chk_all
as
amount number;
begin

 -- 1) V_UTIL_CHK_TABLE_NAMES
 select count(*) into amount from V_UTIL_CHK_TABLE_NAMES;
 if amount>0 then
     p_util_chk_all_print('V_UTIL_CHK_TABLE_NAMES');
 end if;

 -- 2) V_UTIL_CHK_COLUMN_NAMES
 select count(*) into amount from V_UTIL_CHK_COLUMN_NAMES;
 if amount>0 then
     p_util_chk_all_print('V_UTIL_CHK_COLUMN_NAMES');
 end if;

 ...
end;
```

Рисунок Г.1 – Фрагмент процедуры-оболочки

На рис. Г.2 показан пример функции, осуществляющей часть операций контроля соблюдения правил именования таблиц БД.

```
FUNCTION          f_util_get_table_name(table_name
sys.dba_tables.table_name%Type)
return sys.dba_tables.table_name%Type
is
res sys.dba_tables.table_name%Type;
begin
  --правило зарезервированных слов, правило энумераторов
  select
decode(replace(TABLE_NAME,'_DICT',''),'SESSIONS','SESSION','FILES','FILE','USERS
','USER', replace(TABLE_NAME,'_DICT',''))
         into res from dual;
  return res;
end;
```

Рисунок Г.2 – Текст функции контроля соблюдения правил именования

На рис. Г.3 показан пример представления, осуществляющего контроль соблюдения правил именования таблиц базы данных.

```
select OWNER, F_UTIL_GET_TABLE_NAME(TABLE_NAME), TABLESPACE_NAME, NUM_ROWS,
REAL_TABLE_NAME
from dba_tables --правило зарезервированных слов
where
--правило единственного числа
(TABLE_NAME like '%S'
     or
     TABLE_NAME like '%DICT' and TABLE_NAME not like '%_DICT' escape '*'
     -- частично правило энумераторов (не может быть объектов,
     -- заканчивающихся на DICT без подчеркивания перед этим суффиксом)
    )
--Исключения (окончаний US во множественном числе не бывает)
and TABLE_NAME not like '%US'
order by owner, table_name
```

Рисунок Г.3 – Пример представления, осуществляющего контроль соблюдения правил именования таблиц

На рис. Г.4 показан пример представления, осуществляющего контроль соблюдения правил именования атрибутов таблиц базы данных.

```
select OWNER, TABLE_NAME, COLUMN_NAME, DATA_TYPE, DATA_LENGTH
from dba_tab_cols d
and (
-------------------------------------------------------------------------------
--правило повторяющихся столбцов
COLUMN_NAME like '%NAME'
and COLUMN_NAME not like F_UTIL_GET_TABLE_NAME(TABLE_NAME)||'_NAME' escape '*'

--Столбцы-исключения (ФИО)
and COLUMN_NAME not in ('FIRST_NAME', 'LAST_NAME', 'PATRONIMIC_NAME')

-------------------------------------------------------------------------------
or
--Правило столбцов-идентификаторов
(select count (*) from dba_tab_cols d2
 where d2.owner=d.owner and d2.TABLE_NAME=d.TABLE_NAME and
       d2.column_name='ID_'||F_UTIL_GET_TABLE_NAME(d.TABLE_NAME))=0
--отсечение таблиц с составными первичными ключами
and (select count(*) from dba_cons_columns c
     where c.table_name=d.TABLE_NAME
     and c.constraint_name = (select constraint_name from dba_constraints
                              where CONSTRAINT_TYPE = 'P' and
                              TABLE_NAME=d.TABLE_NAME and OWNER=d.OWNER)
     )=1
and d.COLUMN_NAME in (select COLUMN_NAME from dba_cons_columns c
                      where c.table_name=d.TABLE_NAME and c.constraint_name =
                           (select constraint_name from dba_constraints
                            where CONSTRAINT_TYPE = 'P' and
                            TABLE_NAME=d.TABLE_NAME and OWNER=d.OWNER)
                      )
```

Рисунок Г.4 – Пример представления, осуществляющего контроль соблюдения правил именования атрибутов таблиц, лист 1

```
or
--Подправило внешних идентификаторов
COLUMN_NAME  like '%'||F_UTIL_GET_TABLE_NAME(TABLE_NAME)||'%'
and COLUMN_NAME not like '%NAME%' and COLUMN_NAME not like '%DATE%'
and COLUMN_NAME <> 'ID_'||F_UTIL_GET_TABLE_NAME(TABLE_NAME)
--Подправило идентификаторов иерархий
and COLUMN_NAME <> 'ID_PARENT_'||F_UTIL_GET_TABLE_NAME(TABLE_NAME)

)
order by owner, table_name, column_name
```

Рисунок Г.4 – Пример представления, осуществляющего контроль соблюдения правил именования атрибутов таблиц, лист 2

На рис. Г.5 показан пример результатов выполнения процедуры контроля выполнения правил именования объектов (не выполнены правила именования двадцати одного атрибута таблиц базы данных).

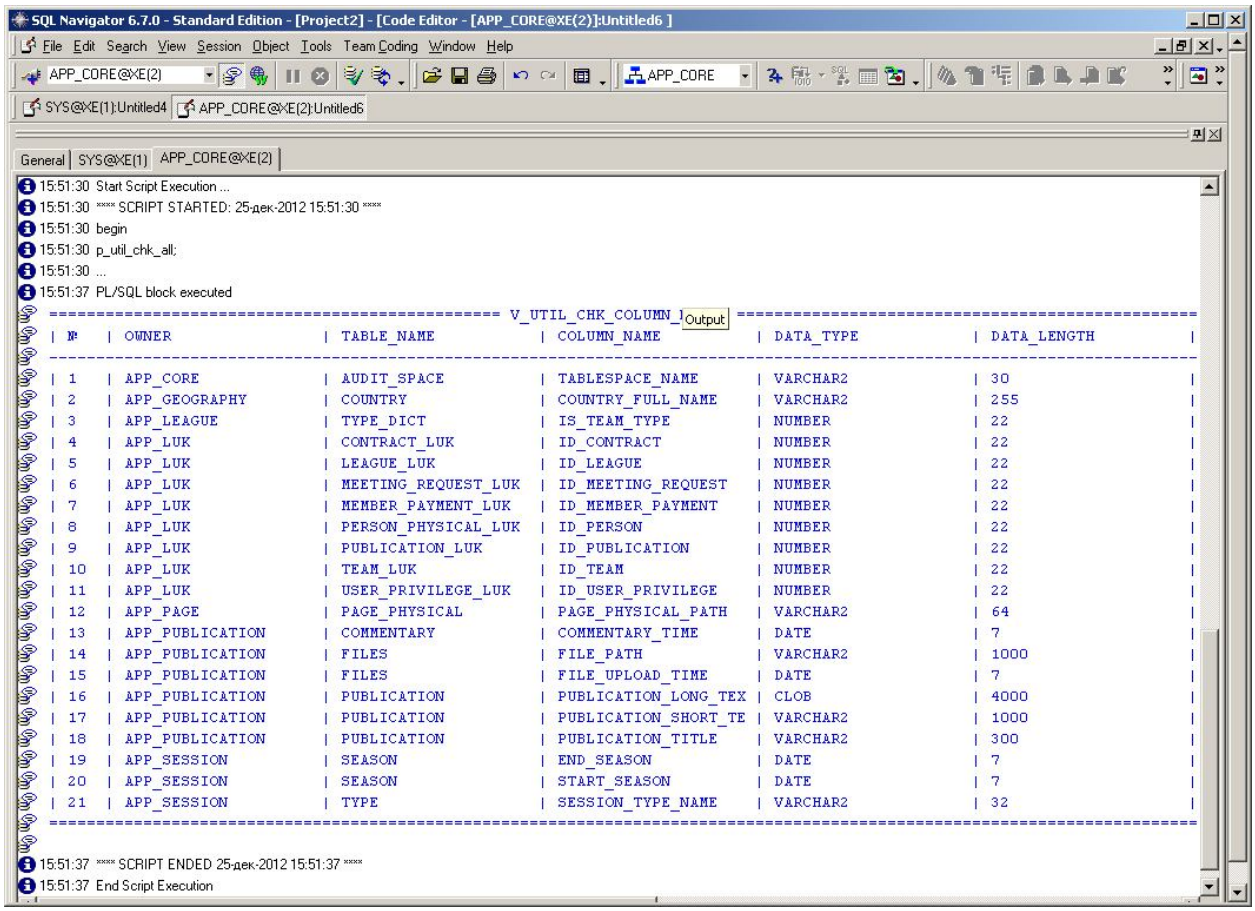

Рисунок Г.5 – Результаты проверки выполнения правил именования столбцов

Приложение Д

ШАБЛОНЫ ТИПОВЫХ ТРИГГЕРОВ, ПРИМЕНЯЕМЫХ ДЛЯ РЕАЛИЗАЦИИ ОГРАНИЧЕНИЙ ЦЕЛОСТНОСТИ

Триггеры, обеспечивающие сюрьективное отображение данных в двух таблицах:

а) для главной таблицы (рис. Д.1);

```
CREATE OR REPLACE TRIGGER t_sync_имя_главной_таблицы_
 AFTER
  INSERT OR DELETE OR UPDATE
 ON имя_главной_таблицы
declare
  amount number;
  amount_owner number;
  offset number;
begin
  select count(*) into amount from имя_главной_таблицы;

  select count(*) into amount_owner from имя_зависимой_таблицы;

  IF INSERTING THEN
      offset := -1;
  ELSIF UPDATING OR DELETING THEN
      offset := 0;
  END IF;

  if  amount + offset <> amount_owner then
      raise_application_error(-20001,'Не все записи таблицы имя_главной_таблицы
('||amount ||') имеют соответствие в таблице имя_зависимой_таблицы
('||amount_owner || ')');
   end if;
end;
```

Рисунок Д.1 – Пример триггера, обеспечивающего сюрьективное отображение данных для главной таблицы

б) для зависимой таблицы (рис. Д.2).

```
CREATE OR REPLACE TRIGGER t_sync_имя_зависимой_таблицы
 AFTER
  INSERT OR DELETE OR UPDATE
 ON имя_зависимой_таблицы
declare
  amount number;
  amount_owner number;
  offset number;
```

Рисунок Д.2 – Пример триггера, обеспечивающего сюрьективное отображение данных для зависимой таблицы, лист 1

```
begin
  select count(*) into amount from имя_главной_таблицы;
  select count(*) into amount_owner from имя_зависимой_таблицы;

  IF INSERTING OR UPDATING THEN
      offset := 0;
  ELSIF DELETING THEN
      offset := -1;
  END IF;

  if  amount + offset <> amount_owner then
      raise_application_error(-20001,'Не все записи таблицы имя_главной_таблицы
('||amount ||') имеют соответствие в таблице имя_зависимой_таблицы
('||amount_owner || ')');
   end if;
end;
```

Рисунок Д.2 – Пример триггера, обеспечивающего сюрьективное отображение данных для зависимой таблицы, лист 2

Триггеры, обеспечивающие уникальность значений комбинаций атрибутов, расположенных в разных таблицах (рис. Д.3). Для независимой таблицы создается триггер с телом, идентичный приведенному.

```
CREATE OR REPLACE TRIGGER t_uniq_имя_зависимой_таблицы
 BEFORE
  INSERT OR UPDATE
 ON имя_зависимой_таблицы
REFERENCING NEW AS NEWROW OLD AS OLD
 FOR EACH ROW
DECLARE
REQ_FLAG INTEGER;
BEGIN
--заменяет UK для полей, расположенных в разных таблицах
SELECT count(*) INTO REQ_FLAG FROM имя_главной_таблицы, имя_зависимой_таблицы
WHERE имя_главной_таблицы.ключевой_столбец =
имя_зависимой_таблицы.ключевой_столбец
and имя_зависимой_таблицы.уникальный_столбец1 = :NEWROW.уникальный_столбец1
and имя_главной_таблицы.уникальный_столбец2 =
     (SELECT имя_главной_таблицы.уникальный_столбец2
      FROM  имя_главной_таблицы
      where  имя_главной_таблицы.ключевой_столбец = :NEWROW.ключевой_столбец);

IF(REQ_FLAG >0) THEN
   RAISE_APPLICATION_ERROR(-20001,'DUPLICATED DATA!');
END IF;
END;
```

Рисунок Д.3 – Пример триггера, обеспечивающего уникальность значений комбинаций атрибутов для независимой таблицы

Триггеры, обеспечивающие контроль внесения данных в дочерние таблицы, когда сильная сущность связана с несколькими слабыми, и данные в слабых сущностях не должны пересекаться между собой:

а) для главной таблицы (рис. Д.4);

```
CREATE OR REPLACE TRIGGER t_splt_имя_главной_таблицы_
 AFTER
  INSERT OR DELETE OR UPDATE
 ON имя_главной_таблицы
declare
  amount number;
  amount_1 number;
  amount_2 number;
  ...
  offset number;
begin
  select count(*) into amount from имя_главной_таблицы;
  select count(*) into amount_1 from имя_зависимой_таблицы_1;
  select count(*) into amount_2 from имя_зависимой_таблицы_2;
  ...

  IF INSERTING THEN
      offset := -1;
  ELSIF UPDATING OR DELETING THEN
      offset := 0;
  END IF;

  if имя_главной_таблицы + offset <> имя_зависимой_таблицы_1 +
имя_зависимой_таблицы_2 + ... then
      raise_application_error(-20001,'Не всем записям в имя_главной_таблицы
('||amount ||') соответствуют записи в таблицах: имя_зависимой_таблицы_1('||
amount_1 ||'), имя_зависимой_таблицы_2 ('|| amount_2 ||'), ...!');
  end if;
end;
```

Рисунок Д.4 – Пример триггера, обеспечивающего контроль внесения данных в главную таблицу

б) для зависимых таблиц (рис. Д.5).

```
CREATE OR REPLACE TRIGGER t_splt_имя_зависимой_таблицы_1
 AFTER
  INSERT OR DELETE OR UPDATE
 ON имя_зависимой_таблицы_1
declare
  amount number;
  amount_1 number;
  amount_2 number;
  split_checksum number;
  offset number;
begin
  select count(*) into amount from имя_главной_таблицы;
  select count(*) into amount_1 from имя_зависимой_таблицы_1;
  select count(*) into amount_2 from имя_зависимой_таблицы_2;
  ...

  select count(*) into split_checksum from (
        select ключевой_столбец from имя_зависимой_таблицы_1
        intersect
        select ключевой_столбец from имя_зависимой_таблицы_2);
  ...
```

Рис. Д.5 – Пример триггера, обеспечивающего контроль внесения данных в зависимую таблицу, лист 1

```
  IF INSERTING OR UPDATING THEN offset := 0;
  ELSIF DELETING THEN offset := -1;
  END IF;
  if amountoffset <> amount_1 + amount_2  + ... then
      raise_application_error(-20001,'Не всем записям в имя_главной_таблицы
('||amount ||') соответствуют записи в таблицах: имя_зависимой_таблицы_1('||
amount_1 ||'), имя_зависимой_таблицы_2 ('|| amount_2 ||'), ...!');
  end if;

  if split_checksum <> 0 then
     raise_application_error(-20001,'В таблицах имя_зависимой_таблицы_1 и
имя_зависимой_таблицы_2 есть пересечения ('||split_checksum||')!');
  end if;
  ...

end;
```

Рисунок Д.5 – Пример триггера, обеспечивающего контроль внесения данных в зависимую таблицу, лист 2

Триггеры, обеспечивающие заполнение (аналог not null option) необязательных полей сильных сущностей, если это требуется для зависящих от них слабых сущностей (рис. Д.6).

```
CREATE OR REPLACE TRIGGER t_null_имя_зависимой_таблицы
 BEFORE
  INSERT OR UPDATE
 ON имя_зависимой_таблицы
REFERENCING NEW AS NEWROW OLD AS OLD
 FOR EACH ROW
DECLARE
FLAG number;
BEGIN
  SELECT decode (nvl(имя_проверяемого_поля,-1),-1,1,0) INTO FLAG
  FROM имя_главной_таблицы
  WHERE ключевой_столбец = :NEWROW.ключевой_столбец;
  IF(FLAG = 1) THEN
      RAISE_APPLICATION_ERROR(-20001,'Таблица имя_главной_таблицы не может иметь
пустое значение в поле имя_проверяемого_поля!');
  END IF;
  …
  END;
```

Рисунок Д.6 – Пример триггера, обеспечивающего заполнение необязательных полей сильных сущностей

Приложение Е

ВЫДЕРЖКИ ИЗ ДОКУМЕНТА «ОТЧЁТ О ПРОВЕДЕННЫХ РАБОТАХ НА ПРЕДПРОЕКТНОЙ СТАДИИ «ФОРМИРОВАНИЕ ТРЕБОВАНИЙ» ПРИ РАЗРАБОТКЕ WEB-БАЗИРОВАННОЙ ИНФОРМАЦИОННО-АНАЛИТИЧЕСКОЙ СИСТЕМЫ «РЕЕСТР ЛУК»

Е.1 Характеристика объекта автоматизации и результатов его функционирования

Объектом автоматизации является Всеукраинская общественная организация «Лига украинских клубов интеллектуальных игр» (ВОО «ЛУК»).

ВОО “Лига украинских клубов интеллектуальных игр” (ЛУК или Лига) создана на Учредительном Съезде в Днепропетровске 2 мая 1998 года, зарегистрирована Министерством юстиции Украины 22 сентября 1998 года (свидетельство о регистрации объединения граждан № 1067).

ВОО «ЛУК» осуществляет свою деятельность согласно действующему законодательству Украины, кодексу Международной ассоциации клубов «Что? Где? Когда?» (МАК), уставу ВОО «Лига украинских клубов интеллектуальных игр» и внутренним нормативно-правовым документам (положения о проведении официальных соревнований, регламент проведения чемпионатов, основные правила интеллектуальных игр и др.).

Основная цель деятельности ВОО «ЛУК» – содействие интеллектуальному развитию граждан Украины и защита гражданских, политических, социальных, экономических, культурных и других прав и свобод своих членов.

Задачи ВОО «ЛУК»:

– содействие распространению среди граждан Украины разнообразных видов интеллектуальной деятельности и интеллектуального творчества;

– содействие налаживанию и развитию контактов и сотрудничества в соответствии с целью деятельности ЛУК в сфере интеллектуального и творческого развития;

– содействие практической реализации идей и предложений, имеющих общественно-полезное значение и отвечающих цели деятельности ЛУК;

– участие в проведении просветительской, научной, культурной, методической, социально-полезной деятельности;

– представление в соответствии с требованиями действующего законодательства Украины интересов членов ЛУК, чьи права нарушены, в органах государственной власти и местного самоуправления, в том числе в судебных учреждениях, а также на международном уровне;

– содействие членам ЛУК в реализации социальных и экономических прав граждан, их защите от нарушений, в том числе со стороны государства;

– обеспечение участия в качественно организованных соревнованиях по интеллектуальным играм максимального количества украинских знатоков;

– повышение конкурентоспособности украинских команд на мировом уровне;

– популяризация интеллектуальных игр в широких слоях населения (прежде всего среди детей, юношества, студентов);

– обеспечение поддержки деятельности ЛУК государственными и коммерческими структурами;

– стимулирование организаторов турниров, повышение организационно-творческого уровня турниров.

Для реализации цели и задач устава ЛУК в установленном порядке осуществляет такие действия:

– координирует деятельность членов ЛУК;

– основывает, согласно требованиям действующего законодательства, собственные средства массовой информации, способствует развитию издательской деятельности;

– распространяет информацию о своей деятельности, пропагандирует идеи и цели ЛУК;

– организовывает и проводит фестивали, конкурсы, форумы, встречи с государственными и общественными деятелями, различные благотворительные акции, творческие встречи, развлекательно-познавательные программы и другие массовые мероприятия с участием членов ЛУК;

– участвует в организации и проведении курсов, лекций, тренингов, семинаров, других просветительских мероприятий для широкой общественности;

– осуществляет поиск и апробацию новых форм игротехники в сфере интеллектуально-творческой деятельности;

– организовывает и проводит Чемпионаты и Кубки Украины по различным видам интеллектуальных игр, устанавливает их регламент, утверждает положения об их проведении и другие действия.

Организационная структура ВОО «ЛУК» приведена в виде схемы на рис. Е.1.

Все решения о назначении членов ЛУК на руководящие должности осуществляются на Съезде ЛУК путём общего голосования. Главой организации является президент ЛУК, в обязанности которого входит общее руководство и координация деятельности ВОО «ЛУК», представление интересов ЛУК, отчётность ЛУК перед госструктурами и т.п. Непосредственно президенту подчиняются модераторы официального и дискуссионного листов ЛУК (выполняют служебные функции организации, поддержки и контроля листов рассылки), председатель ревизионной комиссии ЛУК, который

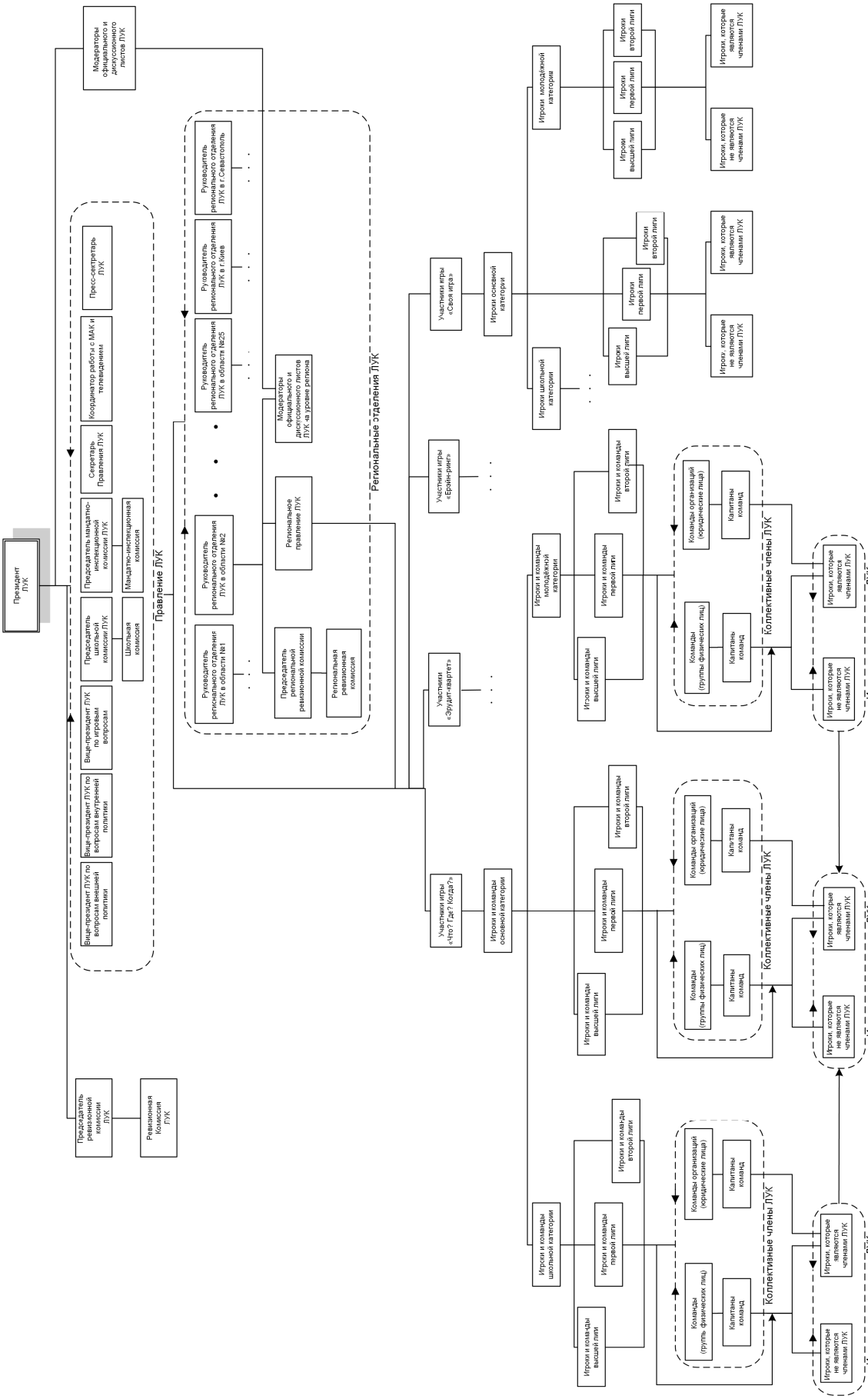

Рисунок Е.1 – Схема организационной структуры Всеукраинской общественной организации «Лига украинских клубов интеллектуальных игр»

возглавляет ревизионную комиссию (осуществляет контроль деятельности президента и Правления, контроль финансовой деятельности организации), и Правление ЛУК, которое в свою очередь включает 8 должностных лиц:

– вице-президент ЛУК по вопросам внешней политики (взаимоотношения с различными государственными и негосударственными структурами, СМИ);

– вице-президент ЛУК по вопросам внутренней политики (взаимоотношения с регионами, членские взносы, реестр ЛУК);

– вице-президент ЛУК по игровым вопросам (контроль проведения чемпионатов и кубков Украины по интеллектуальным играм);

– председатель школьной комиссии ЛУК, который возглавляет школьную комиссию (контроль проведения школьных турниров);

– председатель мандатно-инспекционной комиссии ЛУК, который возглавляет мандатно-инспекционную комиссию (контроль допуска игроков к участию в турнирах, контроль переходов игроков между командами и т.д.);

– секретарь Правления ЛУК (ведение отчётной документации организации);

– координатор работы с МАК и телевидением (взаимодействие с международной ассоциацией клубов (МАК), телекомпанией «Игра», различными телевизионными проектами, связанными с интеллектуальными играми);

– пресс-секретарь ЛУК (взаимодействие с прессой, решение проблем с легализацией игр, организация проведения всеукраинского синхронного турнира).

Правлению ЛУК подчиняются региональные отделения ЛУК в Украине (в 25 областях Украины и в таких приравненных к областям городах, как Киев и Севастополь), которые представлены региональными руководителями. В региональных отделениях присутствуют модераторы официального и дискуссионного листов ЛУК (которые также подчиняются главным модераторам ЛУК), председатель ревизионной комиссии ЛУК (вместе с самой ревизионной комиссией) и Правление ЛУК, которые действуют на региональном уровне.

Дальнейший уровень иерархии в организационной структуре ВОО «ЛУК» представлен членами ЛУК, которые непосредственно принимают участие в играх, не занимая руководящих должностей. Игроки, которые участвуют в турнирах всеукраинского значения, подчиняются непосредственно Правлению ЛУК.

По виду проводимые интеллектуальные игры разделяют на четыре вида: «Что? Где? Когда?», «Брэйн-ринг», «Эрудит-квартет» (возможно как индивидуальное, так и командное участие) и «Своя игра» (возможно только индивидуальное участие).

В рамках соревнований по каждой из игр данного вида все игроки или команды делятся по возрастным категориям, где присутствуют: школьная и молодёжная (студенческая) категории, в которых существует строгое

ограничение по возрасту, а также основная категория, в которой нет ограничений по возрасту (играть в данной категории могут игроки всех перечисленных выше категорий).

В рамках каждой возрастной категории выделяют 3 лиги, которые условно находятся в вертикальной иерархии: высшая, первая и вторая лиги соответственно. Один игрок или команда может состоять одновременно только в одной из данных лиг. По результатам проведения чемпионатов команды получают право перехода в другую лигу: лигу, которая стоит выше в иерархии (если команда показала хорошие результаты); или же переходят в лигу, которая стоит ниже в иерархии (если команда показала плохие результаты).

Существует два вида команд: команды, состоящие из группы физических лиц, и команды организаций, т.е. спонсируемые юридическими лицами (как правило, они состоят из членов данной организации, предприятия и т.д.). У каждой команды есть свой капитан, которому подчиняются остальные члены команды, которые, в свою очередь, делятся на игроков, являющихся членами ЛУК, и игроков, не являющихся членами ЛУК.

Команда, участвующая в турнире, должна иметь основной состав в 6 человек (существует также запасной состав команды), 4 из которых должны являться членами ЛУК. По результатам турниров формируется рейтинг результатов команд или игроков (в зависимости от типа турнира – командный или персональный). Возможно также фестивальное (внезачётное) участие в игре.

Анализируя организационную структуру ВОО «ЛУК», приведенную на рис. Е.1, с точки зрения ИАС «Реестр ЛУК», можно выделить конкретных пользователей системы, к которым относятся все игроки и члены ЛУК, а также другие лица, заинтересованные в получении сведений о результатах деятельности ЛУК через сеть Интернет. Поскольку игроки и команды, а соответственно и члены ЛУК, связаны с определёнными игровыми регионами, привилегированными пользователями системы будут члены региональных правлений ЛУК в 25 областях Украины и в таких городах, как Киев и Севастополь, где существуют городские правления ЛУК, приравненные по статусу к региональным. Региональное правление уполномочено вести финансовую деятельность своего региона, в том числе непосредственно относящуюся к учёту членов ЛУК, учёту оплаты членских взносов, планированию и проведению региональных мероприятий и т.д.

Ещё одним пользователем системы является секретарь Правления ЛУК, который обрабатывает данные для учета и координации деятельности должностных лиц Лиги. Руководящие должности ЛУК могут занимать только её члены, но один и тот же человек может совмещать должности, т.е. занимать сразу несколько должностей.

Согласно уставу ВОО «ЛУК», основатели ЛУК автоматически становятся её членами. Членами Лиги могут быть индивидуальные члены – граждане Украины, иностранные граждане и лица без гражданства, достигшие 14

(четырнадцати) лет, а также коллективные члены – трудовые коллективы предприятий, учреждений, организаций всех форм собственности.

Прием членов осуществляется Правлением ЛУК на основании письменного заявления и после уплаты вступительного взноса. Для вступления в ЛУК лица, желающего стать индивидуальным членом, подается заявление на имя Президента ЛУК, а для вступления в ЛУК коллективного члена предоставляется решение собрания коллектива предприятия, учреждения, организации, которое одновременно является заявлением на вступление в ЛУК. Заявление подается либо непосредственно в Правление, либо в региональное Правление ЛУК, которое передает ее Правлению. Заявление должно быть рассмотрено Правлением на его ближайшем заседании. По результатам рассмотрения поданного заявления Правление выносит решение о принятии в члены ЛУК или об обоснованном отказе в этом.

Лица перестают быть членами Лиги в следующих случаях:

– в случае выхода из состава членов Лиги по собственному желанию (решение принимается Правлением Лиги на основании письменного заявления лица);

– в случае непринятия участия в деятельности Лиги, полной потери связи с ней в течение года и более (членство прекращается автоматически, если от лица в этот срок не поступило заявление с просьбой сохранить его членство в Лиге и объяснением причин потери с ним связи);

– в случае грубого или систематического нарушения требований Устава ЛУК (прекращение членства осуществляется по решению Съезда Лиги, которое принимается не менее, чем 2/3 голосов всех членов, присутствующих на Съезде);

– в случае совершения умышленных действий, причинивших вред Лиге.

Существует два вида членских взносов: вступительные (которые единожды вносятся при вступлении в члены ЛУК) и членские (которые вносятся ежегодно на протяжении всего членства). Размеры взносов зависят от возрастных категорий и от вида членского взноса (взнос индивидуального лица или взнос лица, являющегося частью коллективного члена ЛУК).

Оценкой качества функционирования объекта является эффективность проведения чемпионатов и кубков Украины по различным видам интеллектуальных игр, которая оценивается игроками с точки зрения доступности турниров, предупредительного информационного оповещения, адекватного и справедливого судейства и т.д.

ВОО «ЛУК» как организация взаимодействует с налоговой службой Украины, спонсорами проведения игр и арендодателями помещений, используемых для организации турниров, а также средствами СМИ, организациями и предприятиями, являющимися коллективными членами ЛУК.

Тенденции дальнейшего развития организации направлены на обеспечение самоокупаемости турниров, расширение допустимого количества участников в

турнире и объёмов проводимых игр, дальнейшую популяризацию движения, привлечение большего количества новых игроков, а также стимулирование вступления в члены ВОО «ЛУК».

Е.2 Описание существующей информационной системы

Учёт игроков, команд, переходов игроков между командами, внесение взносов и расчёты результатов проведения турниров ведутся в среде Microsoft Office Excel.

Заявки игроков и команд на участие в турнирах, заявления игроков на вступление в члены ВОО «ЛУК» хранятся организацией в бумажном виде.

Информационная рассылка о предстоящих играх, подтверждении заявок на участие и т.п. проводится с использованием электронной почтовой рассылки. Наряду с электронной почтовой рассылкой для массового оповещения используется web-сайт организации «ЛУК» (www.luk.org.ua), реализующий функции CMS-системы (система WordPress).

К документам, являющимся источником информации для задач ИАС «Реестр ЛУК», относятся:

- заявление на имя Президента ЛУК о вступлении в члены ЛУК от индивидуальных игроков;
- решение собрания коллектива предприятия, учреждения, организации, которое одновременно является заявлением о вступлении в члены ЛУК от каждого индивидуального игрока;
- квитанция об оплате членского взноса за определённый период (полугодие и год);
- членский билет ЛУК, который выдаётся новому члену ЛУК.
- решение Съезда ЛУК об избрании лиц на замещение должностей в ЛУК (протокол голосования);
- заявки команд и игроков на участие в турнирах и чемпионатах ЛУК;
- протоколы результатов проведения турниров и чемпионатов ЛУК и т.д.

Члены региональных правлений получают перечень размеров членских взносов в зависимости от возрастных категорий, которые назначает Правление ЛУК.

Член регионального правления принимает заявление о вступлении в индивидуальные члены ЛУК или решение собрания коллектива организации о коллективном вступлении в ЛУК, соответственно. Эти данные передаются Правлению ЛУК, и после получения решения о принятии заявления Правлением ЛУК новый член ЛУК должен уплатить вступительный взнос, что подтверждается квитанцией об уплате взноса. Вместе со вступительным взносом сразу может быть оплачено и членство за определённый период (полугодие или год) в соответствии с возрастной категорией игрока и типом

членства (индивидуальное, групповое), что также сопровождается квитанцией об оплате. После осуществления данных процедур представитель регионального правления выдаёт новому члену ЛУК членский билет, фиксируя данные в списке членов ЛУК (документ Microsoft Excel).

При получении квитанции об уплате членского взноса за последующие периоды членство продлевается, что фиксируется в документе Excel и в членском билете. За неуплату взносов возможно исключение из членов ЛУК, в случае восстановления членства игроку выдаётся новый членский билет.

Для проведения и организации турниров и чемпионатов из членов ЛУК и представителей регионального или Всеукраинского правления ЛУК оперативно формируются группы, ответственные за их проведение.

Правление ЛУК в лице секретаря Правления формирует перечень руководящих должностей ЛУК, утверждённый членами Правления ЛУК, и хранит в виде электронных документов оперативную информацию о том, кто из членов ЛУК какую должность занимает. После решения Съезда ЛУК о назначении лица на определённую должность секретарь Правления ЛУК фиксирует данные о новом составе руководства.

Е.3 Описание недостатков существующей информационной системы

Используемая для решения главных задач ВОО «ЛУК» среда Microsoft Office Excel, а также хранение и обработка информации в бумажном виде имеет следующие недостатки:

– обязательное присутствие игрока в представительском центре для оформления/подачи заявления о вступлении в члены ЛУК или предварительной заявки на участие в турнире;

– сложность и трудоёмкость проверки заявок на участие при допуске игроков к турниру;

– невозможность просмотра оперативной информации о результатах турниров и отслеживания своей истории игры каждым игроком;

– сложность проведения анализа результатов прошедших турниров для сравнения и прогнозирования игр;

– сложность учета переходов игроков между командами (игрок имеет право осуществлять переход из команды в команду не чаще одного раза в год);

– сложность и низкая оперативность обработки информации (невозможность формирования аналитических выборок, статистических отчётов и т.д.), сложность централизованной аналитической обработки данных;

– отсутствие автоматического оповещения игроков о проводимых играх, о принятии заявок на игру, о скором сроке истечения членства, о просрочке внесения членских взносов, об исключении из членов ЛУК и т.д.;

– сложность ведения финансового учёта членских взносов и контроля членства игроков по результатам уплаты взносов;

– отсутствие автоматического согласования данных между информационным порталом и учетными документами организации.

Е.4 Обоснование необходимости усовершенствования существующей информационной системы объекта

Исходя из описаний объекта управления и существующей информационной системы, а также обнаруженных недостатков существующей системы учета и обработки данных, можно сделать вывод о необходимости качественного и количественного совершенствования существующей информационной системы объекта автоматизации.

Качественное совершенствование системы управления заключается в переходе от традиционных способов управления к способам, ориентированным на широкое использование современных информационных технологий. Такой переход, в свою очередь, вызывает необходимость разработки web-базированной информационно-аналитической системы (ИАС) "Реестр ЛУК", основная цель которой заключается в обеспечении оперативного учёта и анализа хранимых данных.

Количественное совершенствование системы управления означает охват автоматизацией большего количества задач управления. Новые автоматизированные задачи учёта и планирования должны взаимодействовать посредством обмена данными. Кроме того, существующая информационная система должна совершенствоваться для осуществления тенденций дальнейшего развития организации и достижения её целей, которые уже были упомянуты в подразделе Е.3.

Е.5 Цель, критерии и ограничения создания информационно-аналитической системы

Целью создания ИАС является повышение эффективности работы ВОО «ЛУК» путём автоматизации функциональных задач ИАС, что обеспечивает снижение временных затрат организаторов и участников игр на поиск нужной информации; автоматизацию документооборота организации; формирование статистических отчётов для анализа и планирования игр; а также организация web-доступа к актуальной информации о деятельности ЛУК и её участников

для повышения оперативности взаимодействия участников и организаторов игр.

Критерии создания ИАС:

– временной критерий (значительное снижение времени на обработку информации за счет её автоматизации и отсутствие необходимости специального обучения пользователей системы, ответственных за проведение игр);

– стоимостный критерий (стоимость ИАС должна удовлетворять требованиям и возможностям заказчика);

– функционально-стоимостный критерий (стоимость ИАС должна соответствовать объему реализованной в системе функциональности);

– критерий эффективности (ИАС должна быть гибкой, масштабируемой и легко модернизируемой с целью расширения её функциональности и повышения числа пользователей системы);

– критерий актуальности данных (повышение достоверности и актуальности данных в системе, устранение ошибок в данных, характерных для бумажного документооборота, за счет дружественного web-интерфейса пользователя, автоматической валидации вносимых данных и отсутствия запаздывания отражения в системе оперативных данных).

Для ИАС выделяют следующие ограничения:

– ИАС должна обеспечивать корректное отображение web-интерфейса во всех популярных браузерах (Firefox, Internet Explorer, Opera, Google Chrome) без использования платного ПО;

– при разработке ИАС должно быть предусмотрено использование существующих каналов связи;

– ИАС должна обладать высокой производительностью и обеспечивать минимальное время отклика на запросы пользователя;

– ИАС должна разрабатываться с использованием ПО, находящегося в свободном доступе.

Е.6 Функции и задачи создаваемой автоматизированной системы

ИАС «Реестр ЛУК» состоит из трех основных подсистем, охватывающих основные функции ВОО «ЛУК»:

– подсистема организации мероприятий ЛУК;

– подсистема учёта и организации рекламной деятельности;

– подсистема финансовой отчётности.

Наиболее крупной подсистемой ИАС «Реестр ЛУК», подлежащей первоочередной автоматизации, является подсистема учёта и анализа

результатов мероприятий ЛУК, которая включает в себя следующие функциональные задачи (с учётом очерёдности их разработки и внедрения):

- задача «Учёт игроков и команд» (1 очередь);
- задача «Учёт членов ЛУК» (2 очередь);
- задача «Планирование игр» (2 очередь);
- задача «Учёт заявок на игру» (2 очередь);
- задача «Учёт результатов игр» (3 очередь);
- задача «Анализ результатов игр» (4 очередь);
- задача «Учёт объявлений» (4 очередь).

Функциональная структура ИАС «Реестр ЛУК» в целом, а также её отдельных подсистем и задач приведена на рис. Е.2-Е.5.

Задача «Учет игроков и команд» относится к первой очереди автоматизации, поскольку предоставляемые ею данные являются исходными для большинства задач 2-й – 4-й очередей автоматизации. Исходными данными для неё являются заявки капитанов на создание/ликвидацию команды, а также заявки игроков на вступление/уход в команду. Согласно действующим правилам ВОО «ЛУК», игрок не имеет права менять команду чаще 1 раза в год.

Одной из основных задач является задача «Учёт членов ЛУК» (рис. Е.5). Она состоит из таких подзадач:

- подзадача «Учёт членства ЛУК»;
- подзадача «Учёт тарифов вступительных и членских взносов»;
- подзадача «Учёт уплаты вступительных и членских взносов»;
- подзадача «Учёт видов должностей ЛУК»;
- подзадача «Учёт должностей, занимаемых членами ЛУК».

Подзадача «Учёт членства ЛУК» предполагает получение входных данных в виде заявлений на вступление в члены ЛУК или выход из организации, данных об игроках из внешней задачи учёта игроков и данные об оплате членства из подзадачи «Учёт уплаты вступительных и членских взносов». Выходными данными являются данные об игроках, являющихся членами ЛУК, данные о новом члене ЛУК, которые используются в подзадаче «Учёт уплаты вступительных и членских взносов» и другие отчёты и статистические выборки, которые могут понадобиться пользователю задачи.

Подзадача «Учёт тарифов вступительных и членских взносов» предполагает получение входных данных о размерах членских взносов и данных о возрастных категориях. Выходными данными являются данные о текущих тарифах, которые используются в подзадаче «Учёт уплаты вступительных и членских взносов», и другие отчёты и статистические выборки.

Подзадача «Учёт уплаты вступительных и членских взносов» предполагает получение входных данных о текущих тарифах из подзадачи «Учёт уплаты вступительных и членских взносов», данные о новых членах ЛУК из подзадачи

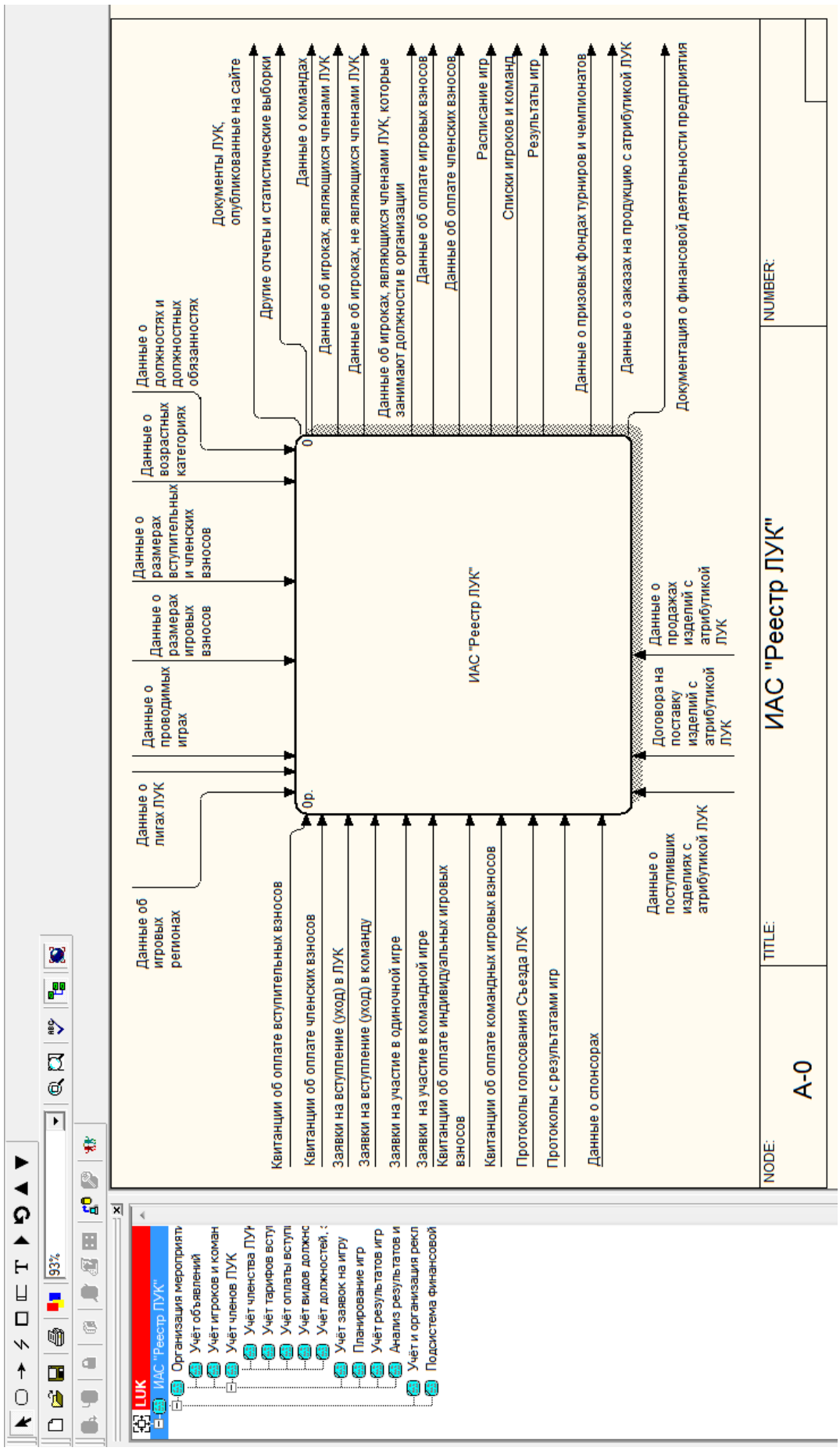

Рисунок Е.2 – Концептуальная диаграмма информационно-аналитической системы «Реестр ЛУК»

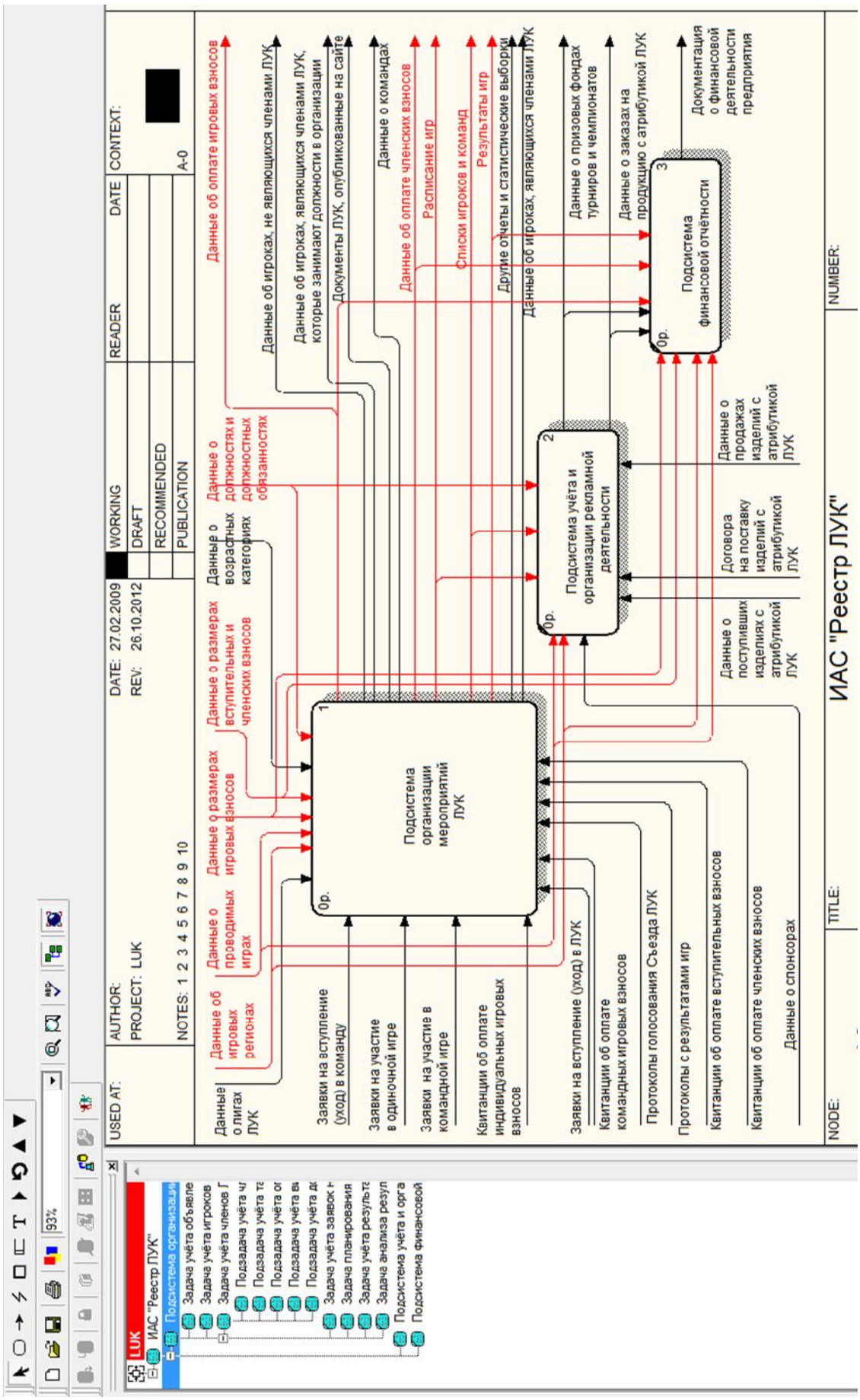

Рисунок Е.3 – Диаграмма первого уровня декомпозиции информационно-аналитической системы «Реестр ЛУК»

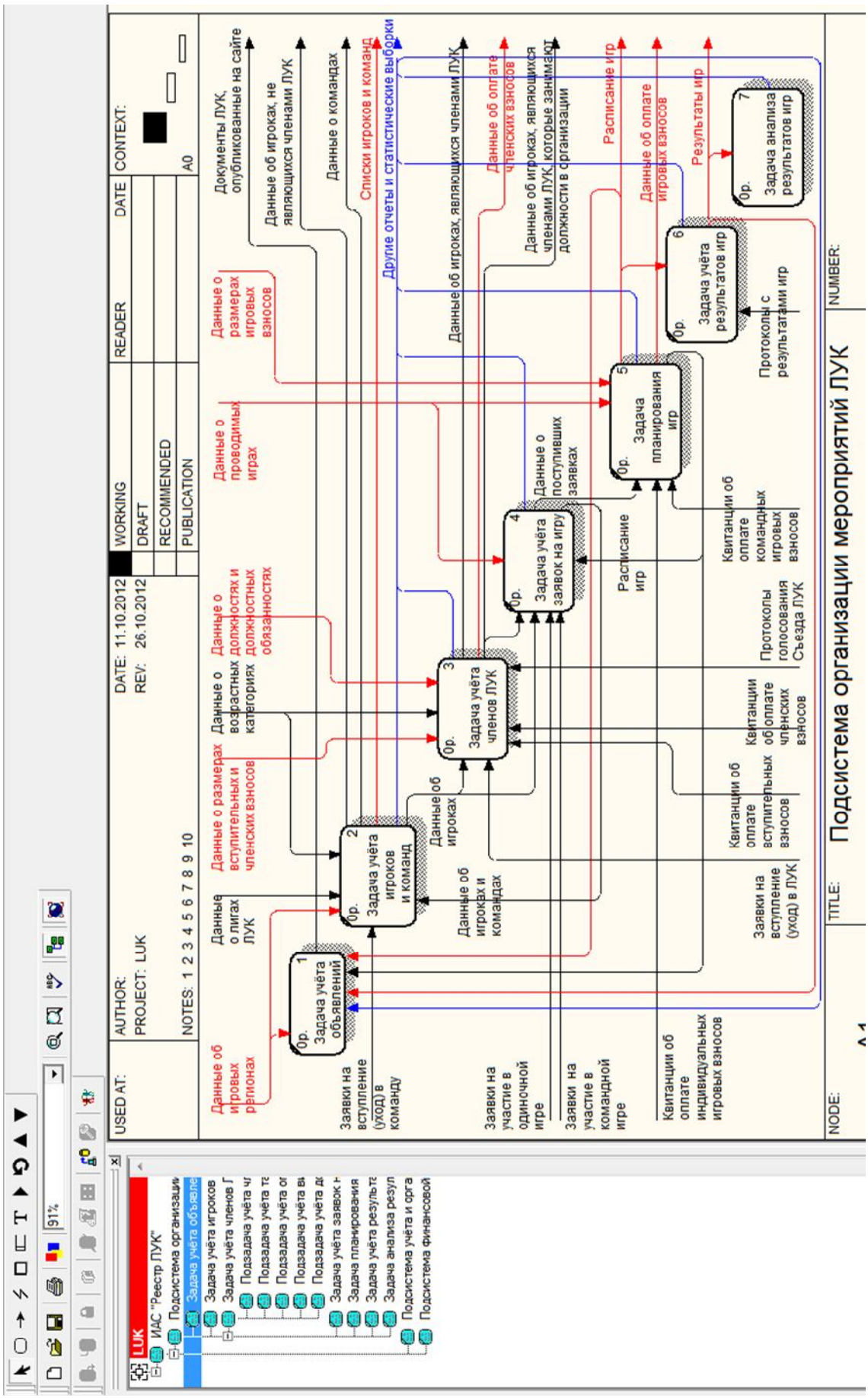

Рисунок Е.4 – Диаграмма второго уровня декомпозиции (подсистема «Организация мероприятий ЛУК»)

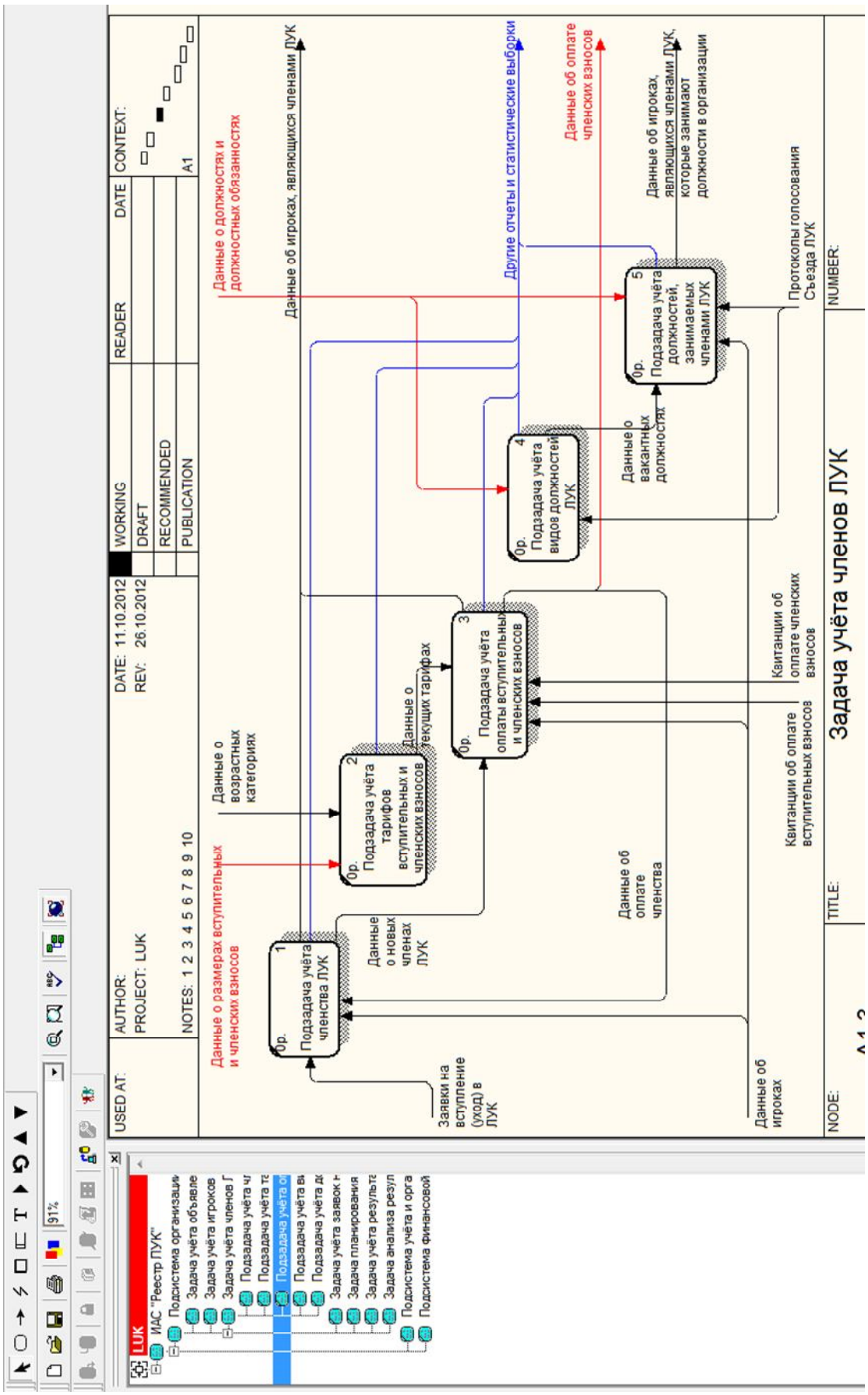

Рисунок Е.5 – Диаграмма третьего уровня декомпозиции (задача «Учёт членов ЛУК»)

«Учёт членства ЛУК», данные об игроках из внешней задачи учёта игроков и данных из квитанций об уплате вступительных и членских взносов. Выходными данными являются данные об игроках, являющихся членами ЛУК, об уплате членских взносов, которые кроме внешней выходной информации задачи являются ещё и входной информацией для задачи «Учёт членства ЛУК», а также другие отчёты и статистические выборки.

Подзадача «Учёт видов должностей ЛУК» предполагает получение входных данных о должностях и должностных обязанностях руководящих лиц и информацию из протоколов голосования Съезда ЛУК о введении/закрытии должностей и изменении должностных обязанностей. Выходными данными являются данные о вакантных должностях, которые передаются в подзадачу «Учёт должностей, занимаемых членами ЛУК», и другие отчёты и статистические выборки.

Подзадача «Учёт должностей, занимаемых членами ЛУК» предполагает получение входных данных о должностях и должностных обязанностях руководящих лиц, об игроках из внешней задачи учёта игроков, о вакантных должностях из подзадачи «Учёт видов должностей ЛУК» и информацию из протоколов голосования Съезда ЛУК о назначении членов ЛУК на руководящие должности. Выходными данными являются данные об игроках, являющихся членами ЛУК, которые занимают должности в организации, и другие отчёты и статистические выборки.

Задача «Планирование игр» обеспечивает ежегодное формирование расписания чемпионатов и отдельных турниров по различным видам игр в различных игровых регионах Украины.

Задача «Учет заявок на игру» обеспечивает предварительный учет заявок, подаваемых командами (включая состав команды на конкретную игру) и игроками дистанционно. Предварительные данные уточняются по факту явки игроков и команд на турниры. Назначением данной задачи является минимизация временных затрат на регистрацию команд и игроков, принимающих участие в турнире, во время проведения мероприятий ЛУК.

Задача «Учет результатов игр» обеспечивает ведение оперативного учета финальных результатов игр на основе формируемых во время турниров в формате Excel протоколов с результатами игр. Указанные протоколы содержат информацию как о промежуточных, так и об окончательных результатах турниров. В перспективе развития ИАС «Реестр ЛУК» возможна дальнейшая автоматизация задач оперативного учета промежуточных результатов игр (статистики ответа команд на игровые вопросы), а также апелляций команд, решений апелляционных комиссий и т.д.

По результатам чемпионатов осуществляется переход команд между игровыми лигами.

Задача «Анализ результатов игр» предполагает формирование рейтингов игроков и команд на основании статистики их выступлений.

Подсистемы учёта и организации рекламной деятельности, а также финансовой отчётности ЛУК решением Правления относятся ко второй и третьей очередям автоматизации соответственно, поэтому в данном документе не рассматриваются.

Описанные функции и задачи были выбраны в соответствии с должностными обязанностями лиц, организующих деятельность ЛУК. Связи между подсистемами, задачами и подзадачами приведены в виде диаграммы потоков данных функциональной структуры ИАС «Реестр ЛУК» на рис. Е.2-Е.5. Данная схема почти идентична схеме функциональной структуры существующей неавтоматизированной системы, так как все выявленные на предпроектной стадии функции подлежат автоматизации.

Е.7 Ожидаемые технико-экономические результаты создания информационно-аналитической системы

В результате создания ИАС «Реестр ЛУК» снизятся затраты на организацию проведения игр (как затраты времени, так и затраты на привлекаемые трудовые ресурсы). За счёт снижения данных затрат и возрастания суммы членских взносов (в результате популяризации организации) возможно улучшение качества проведения интеллектуальных игр и повышение масштабности их проведения (количества одновременно участвующих в турнире команд и игроков).

Ожидаемые затраты на создание и эксплуатацию ИАС планируется компенсировать за счёт членских взносов и передачи разработчику исключительных прав на использование web-базированной ИАС «Реестр ЛУК» как рекламной площадки (ожидаемый трафик 500-1000 посещений информационного портала системы в день). Характер и объем размещаемой рекламы регламентируется договором между разработчиком ИАС и ВОО «ЛУК». При этом ВОО «ЛУК» получает неисключительные права на эксплуатацию системы, что предполагает возможность продажи разработчиком данной системы или её компонентов другому заказчику или их использования при создании новых систем.

Е.8 Выводы и предложения

Разработка ИАС является необходимой для обеспечения эффективной деятельности организации, так как web-базированная информационно-аналитическая система «Реестр ЛУК» предназначена для автоматизации многих рутинных и трудоёмких задач, связанных с работой ВОО «ЛУК». Кроме того, в

настоящий момент на рынке отсутствуют специализированные системы данного класса, учитывающие специфику данной предметной области.

С целью усовершенствования организации и технологии процесса деятельности организации предлагаются:

– заключение договора на сопровождение и хостинг системы;

– планирование бюджета, выделяемого на хостинг, сопровождение, развитие ИАС и оплату аренды используемых доменных имён;

– требование обязательной эксплуатации системы лицами, занимающими должности в организации.

Некоторые рекомендации по созданию автоматизированной системы:

– разрабатываемая система должна иметь сервис-ориентированную архитектуру и обеспечивать взаимодействие с другими системами с использованием формата XML;

– во время внедрения системы рекомендуется осуществить импорт данных из существующей системы (файлов Microsoft Excel);

– рекомендуется использование бесплатной СУБД Oracle XE 11G, поскольку планируемый объём хранимых пользовательских данных не будет превышать максимально допустимый для данной СУБД объем в 11 Гб, а её функциональных возможностей достаточно для реализации требований заказчика в полном объёме;

– в качестве платформы разработки рекомендуется выбрать Java 7 EE, поскольку разрабатываемая система имеет территориально распределённых по разным регионам Украины пользователей и разрабатываемая система должна иметь web-интерфейс;

– в качестве web-сервера рекомендуется выбрать Apache Tomcat 7;

– в качестве среды разработки рекомендуется выбрать Eclipse IDE 4.

Всё перечисленное выше ПО является бесплатным и находится в свободном доступе.

Приложение Ж

ДОКУМЕНТ «ОПИСАНИЕ ТРЕБОВАНИЙ» К ИНФОРМАЦИОННОЙ СИСТЕМЕ «ВИРТУАЛЬНАЯ ДОСКА ОБЪЯВЛЕНИЙ»

Ж.1 Виртуальная доска объявлений. Как я это вижу

Человек хочет что-то приобрести, заходит на сайт и выбирает нужную категорию товара (или задаёт ключевое слово в строке поиска). В начале (при первом отображении главной страницы) на сайте выбор категорий объявлений не показываем, условно полагая, что выбрана "Компьютерная техника".

В этот момент человек может просто просмотреть все расположенные здесь объявления, переходя со страницы на страницу последовательно или выбрать режим фильтрации, указав подраздел, в который попадает искомый товар (в нашем случае, скажем: компьютеры, материнские платы, память, процессоры, видеокарты и т.п.). Может написать письмо автору объявления и предложить свою цену за товар, если цена договорная или объявление подразумевает аукцион (в этом случае возле такого объявления показывается максимальная предложенная цена).

Здесь также должно быть окно регистрации нового пользователя или входа уже зарегистрированного, которое после входа убирается (преобразуется в кнопку «Выход»). Вновь зарегистрированный клиент должен автоматически считаться прошедшим процедуру входа.

Клиент системы получает дополнительные возможности:

- при просмотре объявлений клиент может помещать в свою записную книжку заинтересовавшие его объявления (ставить галочку в поле «Запомнить»);
- сортировать объявления по виду выделения на сайте;
- вывешивать новое объявление на сайте;
- сразу вносить своё предложение о цене, которую он готов заплатить за товар из объявления, где цена договорная. При этом хозяину объявления должно автоматически отправиться письмо с контактными данными этого клиента и предложенная последним сумма. И на сайте последняя предложенная цена должна поменяться автоматически;
- увидеть все существующие свои объявления, с каждым из которых выполнять следующие действия: редактировать, снимать, заказывать продление срока размещения на сайте или варианты выделения объявления, причем заказы выполняются только после оплаты (надо предусмотреть разные варианты оплат, как наличными, так и через безналичный счёт, также и через WEB-money, а в перспективе и через пластиковые карты);

- смотреть свою записную книжку (т.е. выбранные им объявления) с возможностью распечатать их в сокращенном виде (причем или все, или только отмеченные), написать письмо продавцу товара или вступить в иной контакт с ним, а также удалить уже ненужные "записи";

- при согласии на покупку заказывать платную диагностику товара, плата за которую взимается с продавца.

Вот так в общих чертах. Далее конкретнее о мелочах.

Ж.2 Объявления

У каждого объявления должны быть следующие свойства (признаки):

- признак видимости на сайте (действительно или снято. Может быть, стоит его объединить с признаком активности клиента?);
- признак операции («Куплю» или «Продам»);
- признак раздела, в котором оно расположено;
- хозяин объявления;
- дата размещения на сайте;
- дата автоматического удаления с доски, которая вычисляется относительно даты публикации + 14 дней + 3 дня за каждое оплаченное выделение на доске объявлений (причём хозяину должно высылаться сообщение об окончании срока публикации за два дня до окончания);
- признак продления, чтобы хозяин при просмотре всех своих объявлений видел, за какие он уже заплатил, и не удалил их случайно, поскольку деньги возвращаться не будут;
- признак выделения конкретного объявления (размещение на первой странице, выделение шрифтом, выделение цветом, какой-либо динамикой). Неплохо было бы иметь возможность сочетания различных выделений. Причём система должна отслеживать количество оплаченных объявлений на первую страницу, и, если последняя уже заполнена, то не давать заказывать данное выделение или ставить заказ в очередь, указав хозяину, с какого числа его объявление попадёт на первую страницу;
- дату окончания применения выделения (дата оплаты + 3 дня), с отправкой уведомления хозяину, что данная услуга закончена, и о дате снятия его объявления с доски;
- признак занесения объявления в чью-либо записную книжку. Тогда оно не должно физически удаляться из базы, даже если прошёл срок публикации. (Кстати об отображении: объявления в записных книжках клиентов или при просмотре своих объявлений хозяином должны отображаться в соответствии со своим признаком выделения на доске (как при этом выделить объявления с первой страницы доски – надо подумать). Объявления, снятые с доски, хозяин не видит, а в записных книжках клиентов они должны отображаться

"приглушенным" серым цветом;

- признак договорной цены (тогда поле для изменения цены становится доступным для зарегистрированных клиентов).

В объявлении должна содержаться такая информация:

а) что именно предлагается;

б) желаемая или предлагаемая цена (или договорная в случае аукциона);

в) контактный E-mail;

г) контактный телефон.

При публикации объявления необходимо реализовать проверку факта заполнения сведений о контактном E-mail. Заносимая договорная цена должна быть больше предыдущей, иначе изменение цены запрещается.

Ж.3 Регистрация

При регистрации запрашивать ФИО (необязательные поля), логин, пароль, E-mail (обязательные поля) и телефон (необязательно). Проверять на уникальность почтовый адрес. Не допускать регистрацию второго клиента с одного адреса.

Иметь признак активности клиента. Первый раз активацию провести через почту клиента, выслав ему ссылку активации или наш пароль для первого входа. Этим же признаком можно будет в дальнейшем блокировать проштрафившегося клиента, запрещая ему вход на сайт, или снимать с доски все его объявления.

Приложение И

ВИЗУАЛЬНЫЕ МОДЕЛИ ЭЛЕМЕНТОВ ИНТЕЛЛЕКТУАЛЬНОЙ ИНФОРМАЦИОННОЙ ТЕХНОЛОГИИ УСКОРЕННОЙ РАЗРАБОТКИ ИНФОРМАЦИОННЫХ СИСТЕМ И РЕЗУЛЬТАТОВ ЕЕ АПРОБАЦИИ

В данном приложении приведены визуальные модели (схемы данных и диаграммы классов), которые описывают основные проектные решения, применявшиеся в ходе создания интеллектуальной ИТ ускоренной разработки ИС и ее апробации на примерах, рассмотренных в разделе 5.

На рис. И.1 и И.2 рассмотрены фрагменты схемы данных для хранения описаний представлений требований к ИС на уровне информации и для хранения описаний представлений требований к ИС на уровне данных.

На рис. И.3 приведена схема онтологий предметной области в виде сети фреймов, описывающих требования к интеллектуальной ИТ ускоренной разработки ИС.

На рис. И.4 приведена фрагмент схемы онтологии предметной области в виде сети фреймов, описывающих требования к интеллектуальной ИТ ускоренной разработки ИС.

На рис. И.5 приведена схема данных информационно-аналитической системы «Реестр ЛУК», спроектированная с применением традиционного подхода.

На рис. И.6 приведена схема данных информационно-аналитической системы «Реестр ЛУК», спроектированная с применением интеллектуальной ИТ ускоренной разработки ИС.

На рис. И.7 приведена схема фрагмента сети фреймов представления требований к информационно-аналитической системе «Реестр ЛУК» на уровне знаний.

На рис. И.8 приведена схема данных, отражающая паттерны требований, общие для информационно-аналитической системы «Реестр ЛУК» и ИС «Виртуальная доска объявлений».

На рис. И.9 приведена схема данных ИС «Виртуальная доска объявлений».

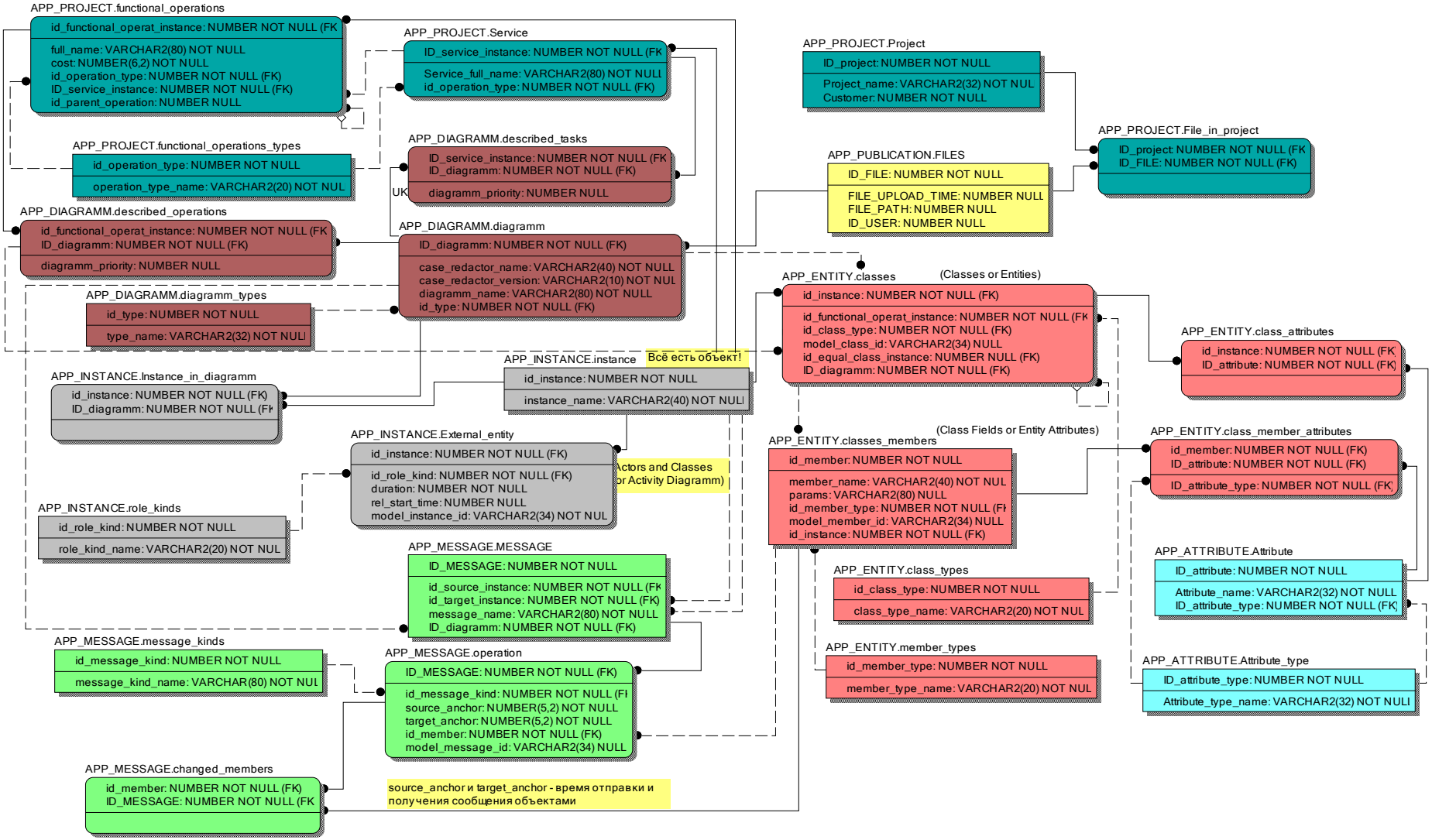

Рисунок И.1 – Диаграмма «сущность – связь» фрагмента схемы данных для хранения описаний представлений требований к информационной системе на уровне информации

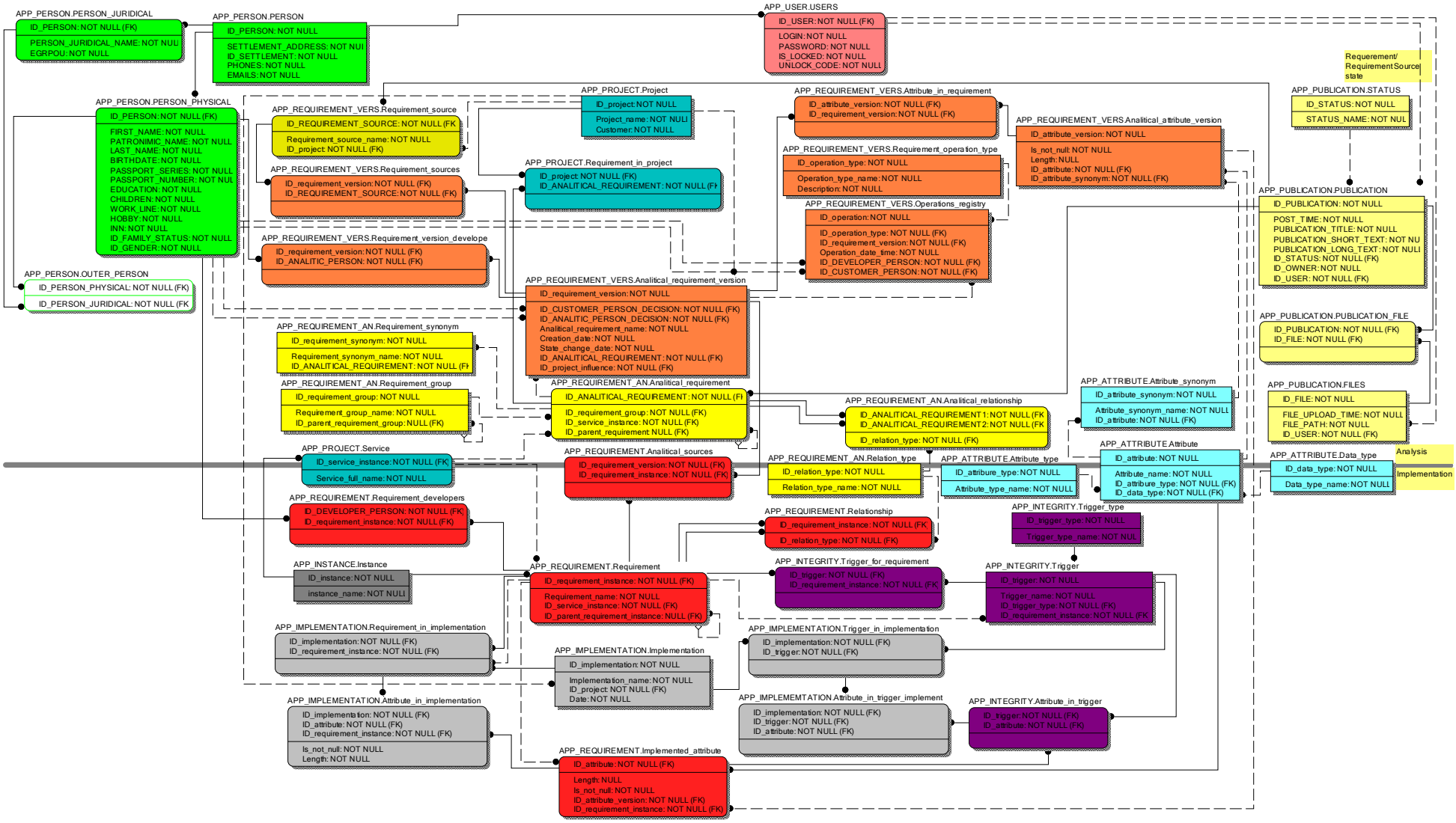

Рисунок И.2 – Диаграмма «сущность – связь» фрагмента схемы данных для хранения описаний представлений требований к информационной системе на уровне данных

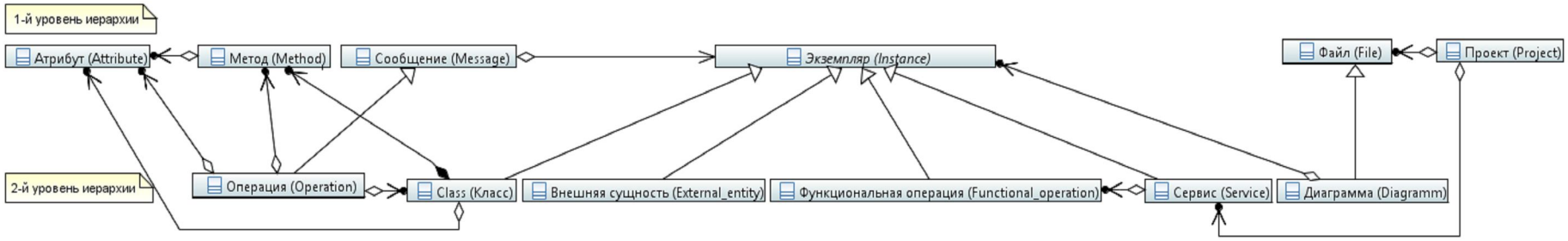

Рисунок И.3 – Онтология предметной области в виде сети фреймов, описывающих требования к интеллектуальной информационной технологии ускоренной разработки информационных систем
(часть 1, идентичная схеме данных на рис. И.1)

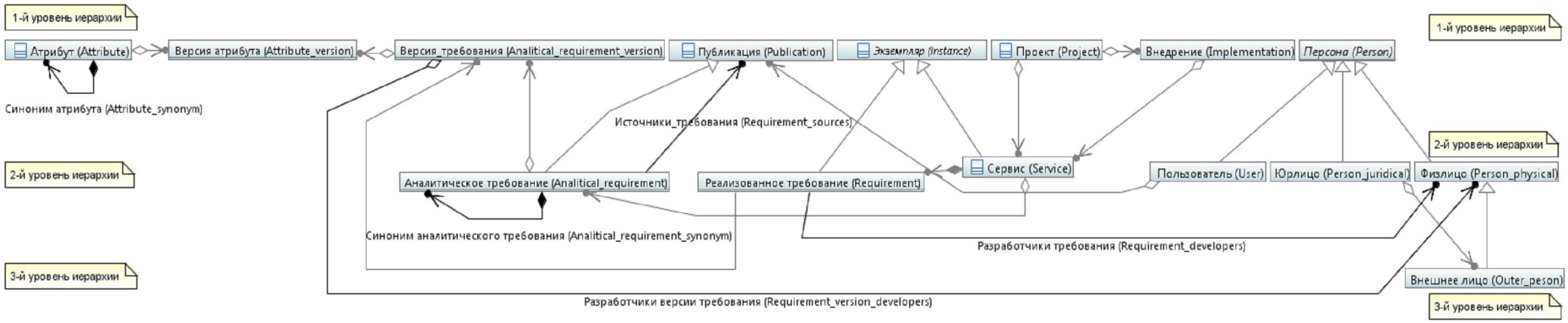

Рисунок И.4 – Фрагмент онтологии предметной области в виде сети фреймов, описывающих требования к интеллектуальной информационной технологии ускоренной разработки информационных систем
(часть 2, идентичная схеме данных на рис. И.2)

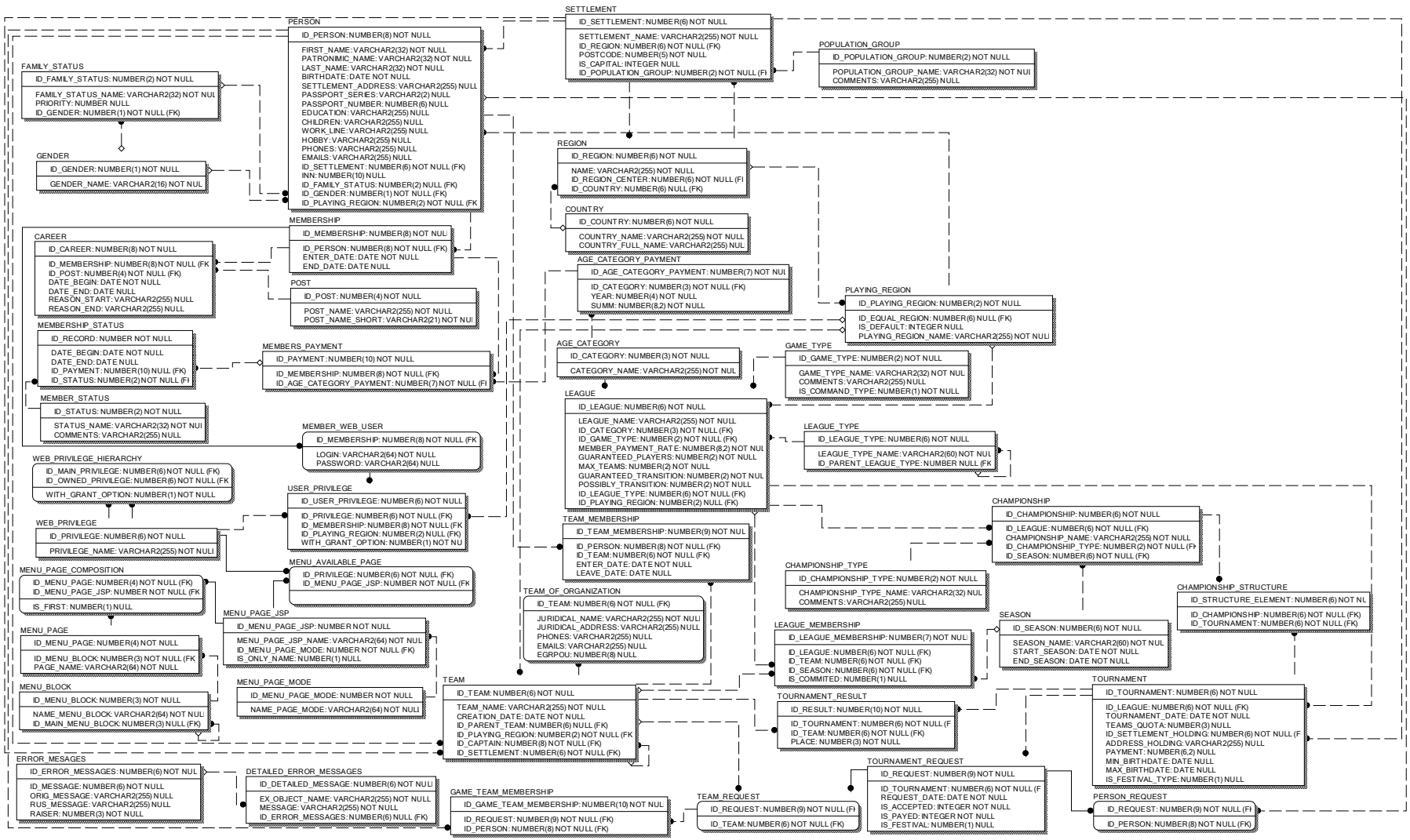

Рисунок И.5 – Схема данных информационно-аналитической системы «Реестр ЛУК», спроектированная с применением традиционного подхода

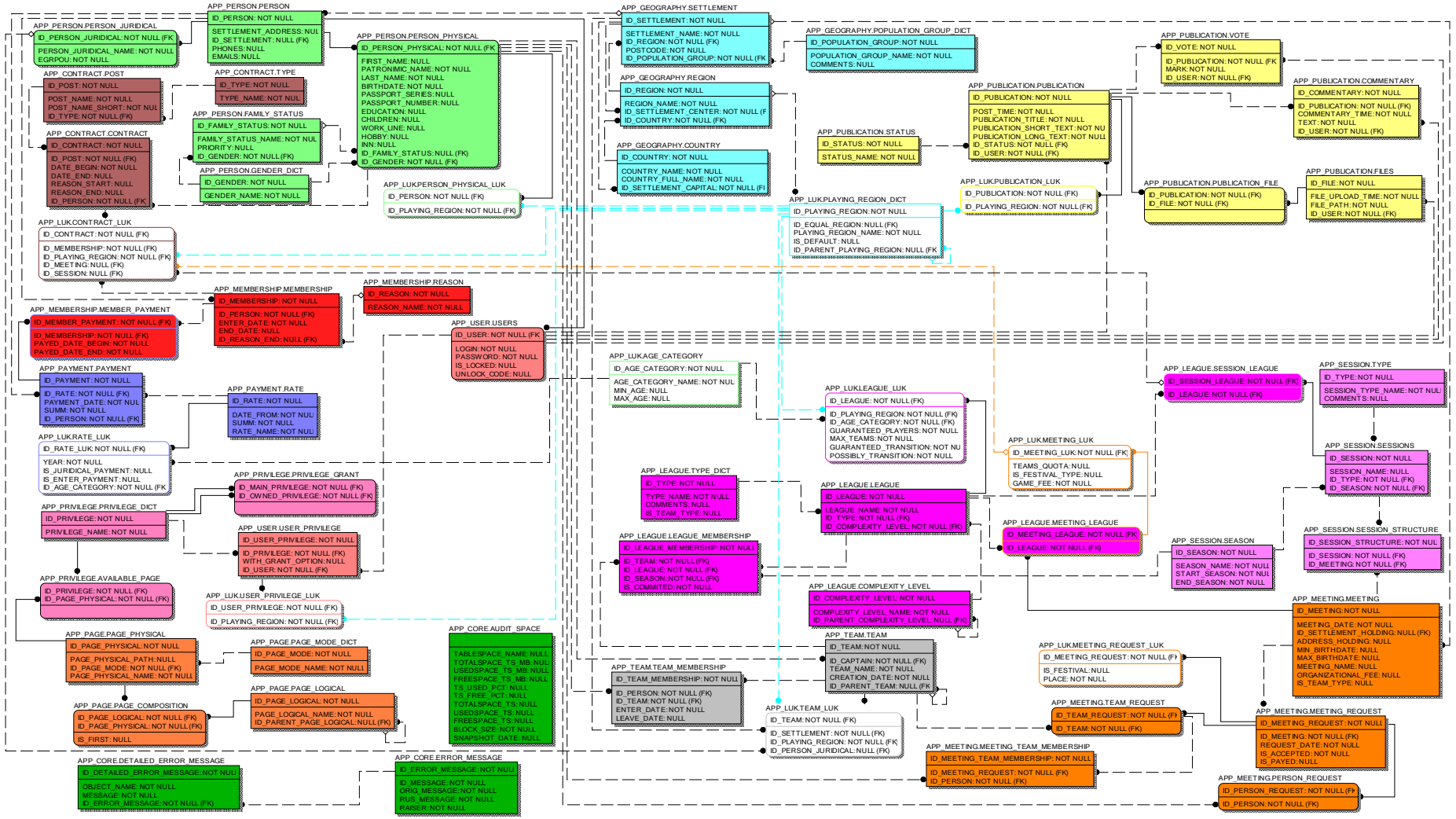

Рисунок И.6 – Схема данных информационно-аналитической системы «Реестр ЛУК», спроектированная с применением интеллектуальной информационной технологии ускоренной разработки информационных систем

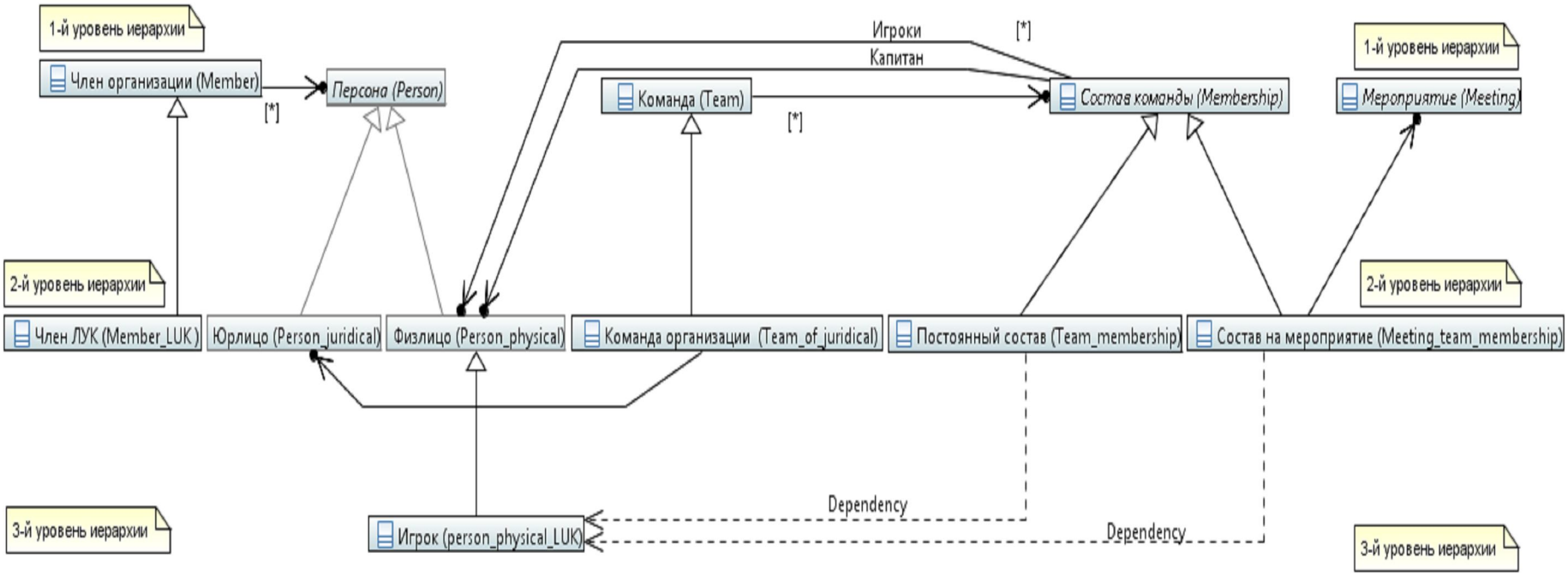

Рисунок И.7 – Фрагмент сети фреймов представления требований
к информационно-аналитической системе «Реестр ЛУК» на уровне знаний

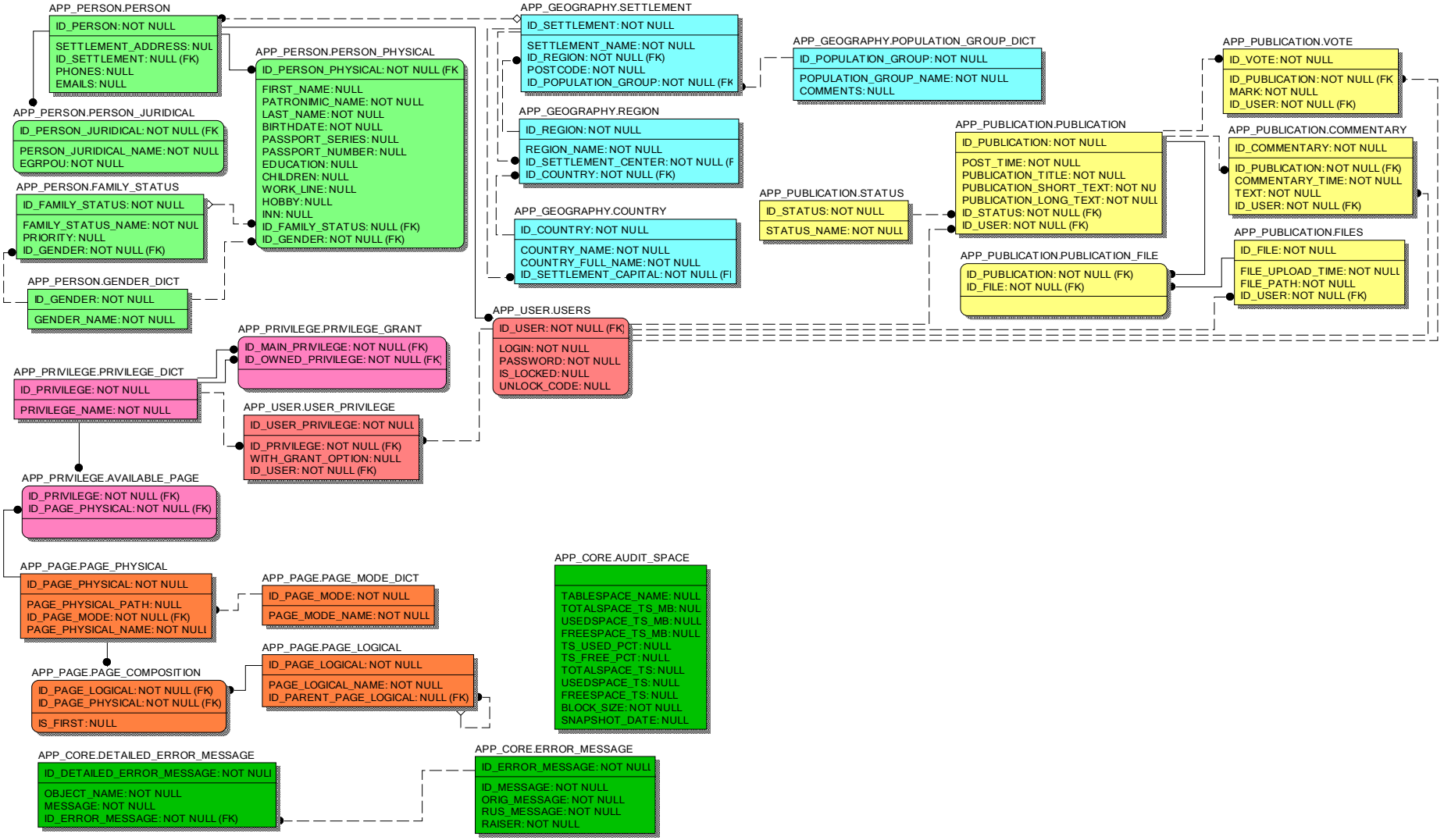

Рисунок И.8 – Схема данных, отражающая паттерны требований, общие для информационно-аналитической системы «Реестр ЛУК» и информационной системы «Виртуальная доска объявлений»

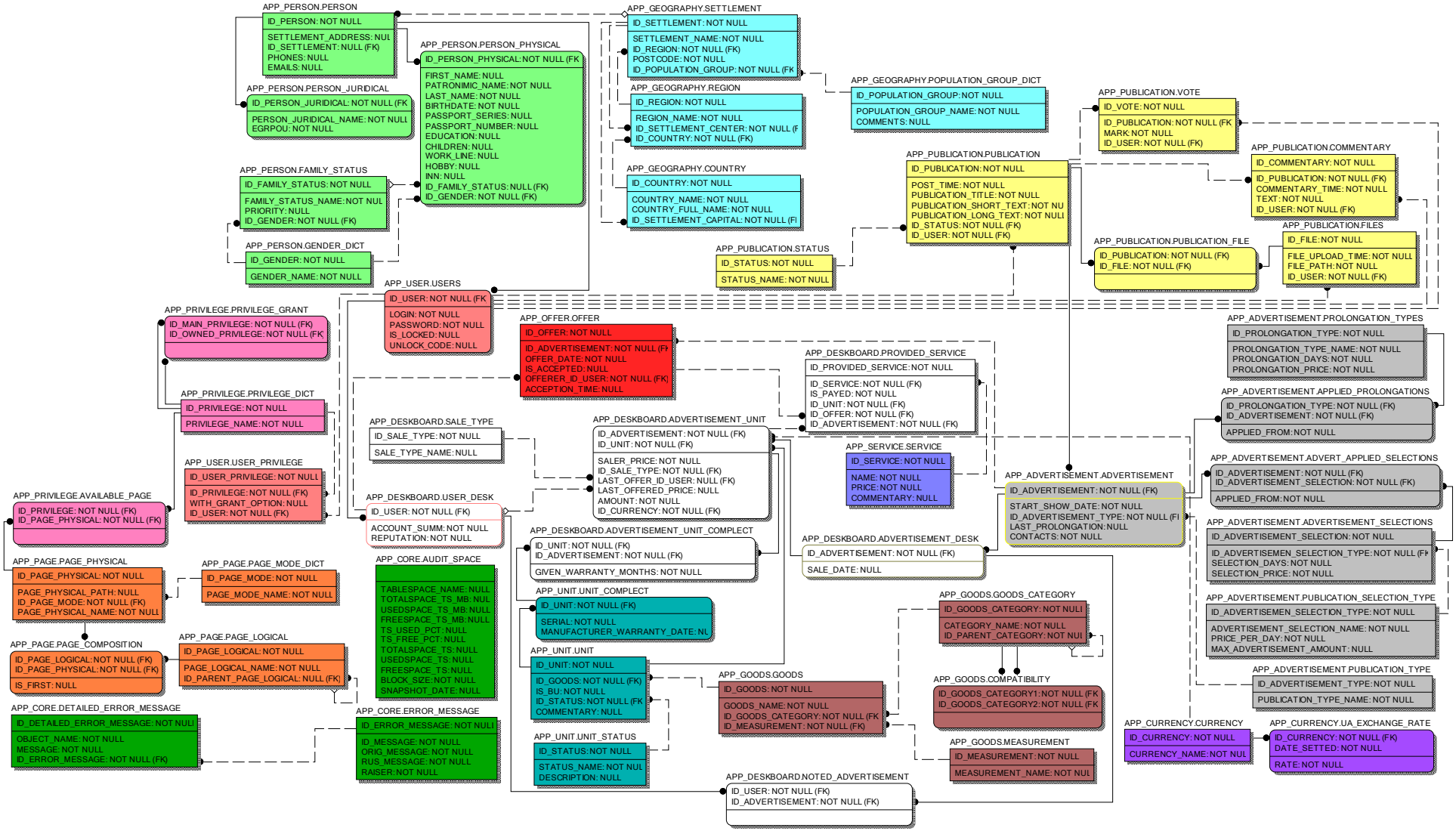

Рисунок И.9 – Схема данных информационной системы «Виртуальная доска объявлений»

ПЕРЕЧЕНЬ ССЫЛОК

1. ГОСТ 34.003–90 Автоматизированные системы. Требования и определения [Электронный ресурс]. – Введ. 01.01.1992. – Режим доступа: http://www.vashdom.ru/gost/34.003-90/.

2. Кузнецов, С.Д. Базы данных: языки и модели: учеб. / С.Д. Кузнецов. – М.: Бином-Пресс, 2008. – 720 с.

3. Чернавский, Д.С. Синергетика и информация (динамическая теория информации) / Д.С. Чернавский. – М. : Едиториал УРСС, 2004. – 288 с.

4. Когаловский, М.Р. Перспективные технологии информационных систем / М.Р. Когаловский. – М. : ДМК Пресс, 2003. – 288 с.

5. An Introduction to Structured Systems Analysis & Design Methodology (SSADM) [Электронный ресурс] / Сайт «Office of Government Chief Information Officer». – Режим доступа: http://www.ogcio.gov.hk/en/infrastructure/methodology/ssadm/doc/s3a_pub.pdf. – Заголовок с экрана.

6. Хазанович, Э.С. Основы проектирования автоматизированных систем управления предприятием / Э.С. Хазанович. – М.: Машиностроение, 1970. – 160 с.

7. Автоматизированная система управления (Теория и методология) / под ред. О.В. Козлова. В 2-х т. – Т.I. – М.: Мысль, 1972. – 455 с.

8. Автоматизированная система управления (Теория и методология) / под ред. О.В. Козлова. В 2-х т. – Т. II. – М.: Мысль, 1972. – 495 с.

9. Автоматизированные системы управления. Применение вычислительной техники и автоматизированных систем управления на предприятиях и отраслях промышленности. – М.: Экономика, 1972. – 399 с.

10. Данильченко, И.А. Проектирование АСУП на основе типовых решений / И.А. Данильченко, А.С. Армясов, В.А. Егорова. – М.: Статистика, 1977. – 213 с.

11. Проектирование и внедрение АСУП / под общ. ред. В.М. Глушкова. – К.: Техніка, 1974. – 191 с.

12. Зайцев, Н.Г. Внедрение и эксплуатация типовой АСУП / Н.Г. Зайцев, М.А. Зорин, Е.А. Чучалов; под общ. ред. В.М. Глушкова. – К.: Техніка, 1976. – 143 с.

13. Общеотраслевые руководящие методические материалы по созданию АСУ предприятиями и производственными объединениями (АСУП). – М.: Статистика, 1977. – 264 с.

14. Справочник разработчика АСУ / под ред. Н.П. Федоренко, В.В. Карибского. – М.: Экономика, 1978. – 583 с.

15. SAP history [Электронный ресурс]. – Режим доступа: www. URL: http://www.sap.com/about/company/history/index.epx. – Заголовок с экрана.

16. Мильнер, Б.З. Теория организации: учеб. / Б.З. Мильнер. – М.: ИНФРА-М, 1999. – 480 с.

17. Мамиконов, А.Г. Проектирование АСУ: учеб. / А.Г. Мамиконов. – М.: Высшая школа, 1987. – 303 с.

18. Адаптивная АСУ производством (АСУ «Сигма») / под ред. Г.И. Марчука. – М.: Статистика, 1981. – 176 с.

19. Глушков, В.М. Основы безбумажной информатики / В.М. Глушков. – М.: Наука, 1982. – 552 с.

20. Свиридов, В.В. Адаптивные интегрированные системы управления производством: учеб. пособие / В.В. Свиридов, В.М. Левыкин, Б.В. Шамша. – Харьков: ХПИ, 1984. – 102 с.

21. Baker, M. The Computer in Manufacturing Unit: One: Computerized Production Control / M. Baker ; Dudley College of Technology. – West Midlands: DCT, 1986. – 50 p.

22. Автоматизированное управление затратами на предприятии / под общ. ред. В. П. Кустарева. – Л.: Машиностроение, 1990. – 227 с.

23. Совершенствование структур управления машиностроительным производством в условиях АСУ / под общ. ред. Л.Ф. Шклярского, А.А. Колобова. – М.: Машиностроение, 1991. – 272 с.

24. Гудушаури, Г.В. Управление современным предприятием / Г.В. Гудушаури, Б.Г. Литвак. – М.: ЭКМОС, 1998. – 336 с.

25. Сергеев, В.И. Логистика в бизнесе / В.И. Сергеев. – М.: ИНФРА-М, 2001. – 608 с.

26. Неизбежность открытых систем // Мир открытых систем и технологий. – 1997. – Вып. 1. – С. 2–10.

27. Ковалев, А. На что потратить 300 000 долларов / А. Ковалев // Компьютеры + Программы. – 1996. – № 8 (32). – С. 4–7.

28. Сипливец, С. Что такое Oracle Applications? / С. Сипливец. В. Фавстов // СУБД. – 1996. – № 2 (40). – С. 6–9.

29. Bancroft, N. H. Implementing SAP R/3: how to introduce a large system into a large organization / N.H. Bancroft. – Greenwich : Manning, 1996. – 266 p.

30. Покровский, Е. R/3 – интегрированная система управления предприятием в среде клиент – сервер / Е. Покровский, М. Аносов // KM-inform. – 1996. – № 7. – С. 18–21.

31. Покровский, Е. Ориентированное на процессы внедрение системы R/3. Ч. 2 / Е. Покровский, М. Аносов // KM-inform. – 1997. – № 4. – С. 21–23.

32. Hammer, M. Reengineering Work: DontXt Automate, Obliterate / M. Hammer // Harvard Business Review. – 1990. – № 7–8. – P. 104.

33. Davenport, T.H. The New industrial Engineering. Information Technology and Business Process Redesign / Thomas H. Davenport, James E. Short // Sloan Management Review. – Cambridge, 1990. – Vol. 31, № 4. – P. 11–17.

34. Hammer, M. How Process Enterprises Really Work / M. Hammer, J. Champy // Harvard Business Review. – 1993. – № 11–12. – P. 108–118.

35. Davenport, T.H. Process Innovation: Reengineering Work through Information Technology / Thomas H. Davenport. – Harvard : Business School Press, 1993. – 338 p.

36. Hammer, M. The Promise of Reengineering / M. Hammer, J. Champy // Fortune. – 1993. – № 9. – P. 94.

37. Хаммер, М. Реинжиниринг корпорации. Манифест революции в бизнесе / М. Хаммер, Дж. Чампи. – СПб. : Изд-во СПбГУ, 1997. – 332 с.

38. Пришвин, А. ERP системы планирования ресурсов предприятия [Электронный ресурс] / А. Пришвин. – Режим доступа: http://www.erpnews.ru/erp.html. – Заголовок с экрана.

39. ERP II [Электронный ресурс]. – Режим доступа: http://erp-expert.com.ua/ERPII. – Заголовок с экрана.

40. Ландэ, Д.В. Поиск знаний в Internet / Д.В. Ландэ. – М.: Диалектика, 2005. – 272 с.

41. Калянов, Г.Н. CASE: структурный системный анализ / Г.Н. Калянов. – М.: Лори, 1996. – 242 с.

42. Вендров, А.М. CASE-технологии. Современные методы и средства проектирования информационных систем / А.М. Вендров. – М.: Финансы и статистика, 1998. – 177 с.

43. Репин, В.В. Процессный подход к управлению. Моделирование бизнес-процессов / В.В. Репин, В.Г. Елиферов. – М.: Стандарты и качество, 2004. – 408 с.

44. Рычков, А. Эволюция ИТ – службы [Электронный ресурс] / А. Рычков // Директор информационной службы. – 2006. – № 9. – Режим доступа: http://www.osp.ru/cio/2006/09/3178211/.

45. Буренин, С. Как нам реорганизовать IT / С. Буренин, М. Литвиновский // Компьютерное обозрение. – 2003. – № 24. – С. 56–58.

46. SOA и EDA: разные архитектуры или одна и та же? [Электронный ресурс]: по материалам BTQmagazine© / пер. А. Маринина. – Режим доступа: http://erpnews.ru/doc2713.html. – Заголовок с экрана.

47. Parikh, A. SOA в реальности [Электронный ресурс] / Ash Parikh, Murty Gurajada; пер. А. Маринина. – Режим доступа: http://erpnews.ru/doc2610.html. – Заголовок с экрана.

48. Захман, Дж. Бизнес и информационные технологи [Электронный ресурс]: лекция / Дж. Захман. – Режим доступа: http://www.intuit.ru/department/itmngt/entarc/1/. – Заголовок с экрана.

49. Luckham, D. The Beginnings of IT Insight: Business Activity Monitoring [Электронный ресурс] / D. Luckham. – Режим доступа: http://complexevents.com/media/articles/cep-article-three.pdf. – Заголовок с экрана.

50. ГОСТ ИСО/МЭК 15288–2005. Системная инженерия. Процессы жизненного цикла систем. – Введ. 01–01–2007. – М. : Стандартинформ, 2006. – 57 с.

51. Архитектура [Электронный ресурс] // Сайт «Большой Энциклопедический словарь». – Режим доступа: http://dic.academic.ru/dic.nsf/enc3p/60458. – Заголовок с экрана.

52. Architecture [Электронный ресурс] // Сайт «Oxford Dictionaries». – Режим доступа: http://oxforddictionaries.com/definition/english/architecture?q=architecture. – Заголовок с экрана.

53. Сайт ISO/IEC/IEEE 42010 Website [Электронный ресурс]. – Режим доступа: http://www.iso-architecture.org/ieee-1471/index.html. – Заголовок с экрана.

54. Zachman, J.A. Architecture is Architecture is Architecture [Электронный ресурс] // Сайт «Zachman International». – Режим доступа: http://test.zachmaninternational.com/index.php/ea-articles/68-architecture-is-architecture-is-architecture. – Заголовок с экрана.

55. Perry, D.E. Software architecture [Электронный ресурс] / D.E. Perry, A.L. Wolf // Режим доступа: http://users.ece.utexas.edu/~perry/work/papers/swa89.pdf. – Заголовок с экрана.

56. Kruchten, Ph. The Rational Unified Process: An Introduction / Ph. Kruchten. – Addison-Wesley Professional, 2004. – 310 p.

57. Helland, P. Metropolis [Электронный ресурс] / P. Helland // Сайт MSDN. – Режим доступа: http://msdn.microsoft.com/en-us/library/aa480026.aspx. – Заголовок с экрана.

58. Veryard, R. Metropolis and SOA Governance [Электронный ресурс] / R. Veryard, Ph. Boxer // Сайт MSDN. – Режим доступа: http://msdn.microsoft.com/en-us/library/aa480051.aspx. – Заголовок с экрана.

59. Holcman, S.B. Driving Efficiency and Innovation by Consistently Managing Complexity and Change [Электронный ресурс] // Сайт MSDN Architecture Center. – Режим доступа: http://msdn.microsoft.com/en-us/architecture/ff476941. – Заголовок с экрана.

60. Enterprise Architecture Definition [Электронный ресурс]. – Режим доступа: http://samvak.tripod.com/earf.pdf. – Заголовок с экрана.

61. Информационная система [Электронный ресурс] // Сайт «Глоссарий.ru». – Режим доступа: http://www.glossary.ru/cgi-bin/gl_sch2.cgi?RIt(uwsg.outt:l!xoxyls:. – Заголовок с экрана.

62. Соммервил И. Инженерия программного обеспечения, 6-е издание: пер. с англ. – М.: Издательский дом «Вильямс», 2002. – 624 с.

63. Aksirt, M. Software Architectures and Component Technology. – Boston, Kluwer Academic Publisher, 2002. – 385 p.

64. Палагин А.В., Кургаев А.Ф. Проблемная ориентация в развитии компьютерных архитектур // Кибернетика и системный анализ. – 2003. – № 4. – С. 167-180.

65. Глушков В.М. Проблемная ориентация и другие пути повышения эффективности ЭВМ // Кибернетика. Вопросы теории и практики. – М.: Наука, 1986. – С. 162-170.

66. Смит К., Уильямс Л. Эффективные решения: практическое руководство по созданию гибкого и масштабируемого программного обеспечения: пер. с англ. – М.: Изд. «Вильямс», 2003. – 448 с.

67. Starke, G Effektive Software-Architekturen: Ein praktischer Leitfaden / Hanser Verlag, Muenchen, 2002. – 432 p.

68. Maciaszek L.A. Requirements Analysis and System Design / L.A. Maciaszek: 2d ed. – Reading: Addison Wesley, Harlow England, 2005. – 504 p.

69. Foegen M., Batterfeld J. Die Rolle der Architektur in der Anwendungsentwicklung // Informatik-Spektrum. Springer. – 2001. - № 5(24). – S. 290-301.

70. Архитектуры, модели и технологии программного обеспечения информационно-управляющих систем: монография / Ткачук Н.В., Шеховцов В.А., Кукленко Д.В., Сокол В.Е. Под ред. М.Д. Годлевского. – Харьков: НТУ «ХПИ», 2005. – 546 с.

71. РД 50–34.698–90. Автоматизированные системы. Требования к содержанию документов [Электронный ресурс]. – Введ. 01.01.1992. – Режим доступа: http://document.ua/kompleks-standartov-i-rukovodjashih-dokumentov-na-avtomatizinor3019.html. – Заголовок с экрана.

72. ГОСТ 34.602–89. Техническое задание на создание автоматизированной системы [Электронный ресурс]. – Введ. 01.01.1990. – Режим доступа: http://vt.ulstu.ru/sites/default/files/%D0%93%D0%9E%D0%A1%D0%A2%2034.602-89_0.pdf. – Заголовок с экрана.

73. SSADM Version 4 Reference Manual. – Oxford: NCC Blackwell, 1990. – 1400 p.

74. Кириллов, В.П. Технология SSADM: методика определения требований к автоматизированной системе / В.П. Кириллов // Компьютеры + Программы, 1994. – № 3 (11). – С. 30–36.

75. Фатрелл, Р.Т. Управление программными проектами: достижение оптимального качества при минимуме затрат / Р.Т. Фатрелл, Д.Ф. Шафер, Л.И. Шафер. – М.: Вильямс, 2003. – 1136 с.

76. Леффингуэлл, Д. Принципы работы с требованиями к программному обеспечению. Унифицированный подход / Д. Леффингуэлл, Д. Уидриг. – М.: Вильямс, 2002. – 448 с.

77. Виггерс, К.И. Разработка требований к программному обеспечению / К.И. Виггерс. – М.: Русская редакция, 2004. – 576 с.

78. Lawrence, B. Unresolved ambiguity: The silent source of risk in your project / Brian Lawrence // American Programmer. – 1996. – Vol. 9, № 4. – P. 18–22.

79. Sommerville, I. Requirements Engineering: A Good Practice Guide / Ian Sommerville, Pete Sawyer. – Chichester : John Wiley & Sons, 1997. – 391 p.

80. Халл, Э. Разработка и управление требованиями: практическое руководство пользователя [Электронный ресурс] / Э. Халл, К. Джексон, Дж. Дик ; пер. с англ. И. Корнипаева. – 2-е изд. – 2005. – Режим доступа: http://80.250.162.180/2002/Yury.Kupriyanov/eBook_RU_Requirements_Engineering.pdf. – Заголовок с экрана.

81. Шрейдер, Ю.А. Системы и модели / Ю.А. Шрейдер, А.А. Шаров. – М.: Радио и связь, 1982. – 152 с.

82. Moore, G.A. Crossing the Chasm: Marketing and Selling High-Tech Products to Mainstream Customers / Geoffrey A. Moore. – New York: HarperBusiness, 1991.

83. Кобёрн А. Современные методы описания функциональных требований к системам / А. Кобёрн. – М.: «Лори», 2002. – 288 с.

84. Макарова, Н.В. Информатика / Н.В. Макарова: учебник. – М.: Финансы и статистика, 2003. – 768 с.

85. Paulk, M. The Capability Maturity Model: Guidelines for Improving the Software Process / M. Paulk et al. – Reading, MA: Addison-Wesley, 1995.

86. Новичков А. Роль процесса Управления Требованиями при разработке сложных программных систем. Практика применения методологии IBM RUP и инструмента IBM Rational RequisitePro [Электронный ресурс] / Сайт компании СМ Консалт. – Режим доступа: http://cmcons.com/articles/upravlenie_trebovanijami_instrument_ibm_rational_r/rol_protsessa_upravlenija_trebovanijami_pri_razrabotke_slozhnykh_programmnykh_sistem_praktika_primenenija_metodologii_ibm_rup_i_instrumenta_ibm_rational_requisitepro/. – Заголовок с экрана.

87. Волков, Ю.О. Управление требованиями и автоматизация этого процесса [Электронный ресурс] / Сайт Юрия Волкова. – Режим доступа: http://yurivolkov.com/articles/Requirements_management_automation_ru.html. – Заголовок с экрана.

88. Деревянко, А.С. Технологии и средства консолидации информации: учеб. пособие / А.С. Деревянко, М.Н. Солощук. – Х. : НТУ "ХПИ", 2008. – 432 с.

89. Евланов, М.В. Глобальные цели поставщика и потребителя ИТ-услуг / М.В. Евланов, О.Е. Неумывакина, А.Ю. Карамышева // Восточно-европейский журнал передовых технологий. – 2012. – № 5/2 (59). – С. 12-17.

90. Web services (application services) [Электронный ресурс]. – Режим доступа: http://searchsoa.techtarget.com/definition/Web-services. – Заголовок с экрана.

91. Cooney, P. Web-services [Электронный ресурс] / P. Cooney. – Режим доступа: http://www.webmascon.com/topics/technologies/webservices.shtml. – Заголовок с экрана.

92. Lande, N. What is Web Service? [Электронный ресурс] / N. Lande. – Режим доступа: http://www.codeproject.com/Articles/17908/What-is-Web-service. – Заголовок с экрана.

93. Web Services – Web Services Tutorials [Электронный ресурс]. – Режим доступа: http://www.roseindia.net/webservices/webservices.shtml. – Заголовок с экрана.

94. What are Web Services [Электронный ресурс]. – Режим доступа: http://www.tutorialspoint.com/webservices/what_are_web_services.htm. – Заголовок с экрана.

95. Основы системного анализа и проектирования АСУ / Под общ. ред. А.А. Павлова. – К.: Выща школа, 1991. – 367 с.

96. ГОСТ 34.601–90. Автоматизированные системы. Стадии создания. – Введ. 01.01.1992. – М. : Изд-во стандартов, 1997. – 10 с.

97. Горобец, Н. ISO 20000: зрелое управление ИТ-услугами / Н. Горобец // Директор информационной службы. – 2006. – № 9. – С. 62–67.

98. ГОСТ Р ИСО/МЭК 20000–200Х (проект, окончательная редакция). Управление услугами. Ч. 1. Спецификация [Электронный ресурс]. – М., 2009. – Режим доступа: http://itsmforum.ru/reference/ISO20000/GOST_R_ISO_MEK_20000-1.pdf. – Заголовок с экрана.

99. ГОСТ ИСО/МЭК 12207–2010. Системная и программная инженерия. Процессы жизненного цикла программных средств. – Введ. 01–03–2012. – М.: Стандартинформ, 2011. – 106 с.

100. Петров, Э.Г. Методология структурного системного анализа и проектирования крупномасштабных ИУС. Ч. 1. Концепции и методы / Э.Г. Петров, С.И. Чайников, А.О. Овезгельдыев. – Харьков: Рубикон, 1997. – 140 с.

101. SSADM V4.2 Structural Standards [Электронный ресурс] / Office of Government Chief Information Officer. – Режим доступа: http://www.ogcio.gov.hk/en/infrastructure/methodology/ssadm/ssadm_42_structural_standards.htm. – Заголовок с экрана.

102. Davis, Alan M. Software Requirements: Objects, Functions and States / Alan M. Davis. – Englewood : Prentice Hall, 1993. – 521 p.

103. Евланов, М.В. Определение понятия «требование к информационной системе» / М.В. Евланов // Вісник Академії митної служби України. Серія «Технічні науки». – 2012. - № 2. – С. 71-77.

104. Рубцов, С.В. Уточнение понятия «бизнес-процесс» / С.В. Рубцов // Менеджмент в России и за рубежом. – 2001. – № 6. – С. 26–33.

105. Бусленко, Н. П. Моделирование сложных систем / Н.П. Бусленко. – М.: Наука, 1978. – 399 с.

106. Aquilano, N.J. Fundamentals of operations management / N.J. Aquilano, R.B. Chase, M.M. Davis. – Chicago : Irwin, 1995. – 667 p.

107. Chase, R.B. Production and operations management: manufacturing and services / R.B. Chase, N.J. Aquilano, R.F. Jacobs. – Boston: Irwin, 1998. – 889 p.

108. Философский энциклопедический словарь / редкол.: Л.Ф. Ильичёв, П.Н. Федосеев, С.М. Ковалёв, В.Г. Панов. – М.: Советская энциклопедия, 1983. – 840 с.

109. Фаулер, М. UML в кратком изложении. Применение стандартного языка объектного моделирования / М. Фаулер, К. Скотт. – М.: Мир, 1999. – 191 с.

110. Bellinger, G. Data, Information, Knowledge and Wisdom [Электронный ресурс] / Gene Bellinger, Durval Castro, Anthony Mills. – Режим доступа: http://www.systems-thinking.org/dikw/dikw.htm. – Заголовок с экрана.

111. Davis, Alan M. Principles of Software Development / Alan M. Davis. – New York: McGraw-Hill, 1995. – 240 p.

112. Васильцова, Н.В. Разработка метамодели требований к информационной системе / Н.В. Васильцова, М.В. Евланов, И.Ю. Панферова // АСУ и приборы автоматики. – 2004. – Вып. 129. – С. 19–27.

113. Евланов, М.В. Концепция представления требований к информационной системе / М.В. Евланов // Информационные системы и технологии: материалы Междунар. науч.-техн. конф., Морское-Харьков, 22-29 сентября 2012 г.: тезисы докладов / редкол.: А.Д. Тевяшев (отв. ред.) и др. – Харьков: НТМТ, 2012. – С. 34.

114. Цаленко, М.Ш. Основы теории категорий / М.Ш. Цаленко, Е.Г. Шульгейфер. – М.: Наука, 1974. – 256 с.

115. Фейс, К. Алгебра: кольца, модули и категории. В 2-х т. – Т. 1. / К. Фейс. – М.: Мир, 1977. – 688 с.

116. Букур, И. Введение в теорию категорий и функторов / И. Букур, А. Деляну. – М.: Мир, 1978. – 259 с.

117. Комраков Б.П. Структуры на многообразиях и однородные пространства / Б.П. Комраков. – Минск: Наука и техника, 1978. – 352 с.

118. Евланов, М.В. Модель методологии предпроектного обследования объекта автоматизации / М.В. Евланов // Збірник тез доповідей науково-практичної конференції «Застосування інформаційних технологій у підготовці та діяльності сил охорони правопорядку». – Харків: Академія Внутрішніх військ МВС України, 2013. – С. 76-78.

119. Левыкин В.М. Задача определения функторов между категорными моделями информационной системы / В.М. Левыкин. М.В. Евланов // Проблемы бионики. – 2003. – Вып. 58. – С. 62-67.

120. Левыкин В.М. Концепция построения CASE-системы разработки информационных управляющих систем / В.М. Левыкин. М.В. Евланов, Мухайрат Мохаммад // АСУ и приборы автоматики. – 2001. – Вып. 114. – С. 55-59.

121. Евланов М.В. Формализация взаимных отображений моделей информационных систем / М.В. Евланов // Materialy IV Miedzynarodowej naukowi-praktycznej konferencji „Nowoczesnych naukowych osiagniec - 2008”. – Tym 13. Matematyka. Fizyka. Nowoczesne informacyjne technologie. – Przemysl: Nauka i studia. – Str. 82-85.

122. Маклаков, С.В. Создание информационных систем с AllFusion Modeling Suite / С.В. Маклаков. – М.: ДИАЛОГ-МИФИ, 2003. – 422 с.

123. Гамма, Э. Приемы объектно-ориентированного проектирования. Паттерны проектирования / Э. Гамма, Р. Хелм, Р. Джонсон, Д. Влиссидес. – СПб.: Питер, 2010. – 366 с.

124. Фримен, Э. Паттерны проектирования / Э. Фримен, Э. Фримен, К. Сьерра, Б. Бейтс. – СПб.: Питер, 2011. – 656 с.

125. Кериевски, Дж. Рефакторинг с использованием шаблонов / Дж. Кериевски. – М.: Издательский дом «Вильямс», 2006. – 400 с.

126. Эмблер, С.В. Рефакторинг баз данных. Эволюционное проектирование / С.В. Эмблер, П.Дж. Садаладж. – М.: Издательский дом «Вильямс», 2007. – 368 с.

127. Фаулер, М. Рефакторинг. Улучшение существующего кода / М. Фаулер. – М.: Символ-Плюс, 2008. – 432 с.

128. Refactoring [Электронный вариант] / Сайт «SourceMaking». – Режим доступа: http://sourcemaking.com/refactoring. – Заголовок с экрана.

129. Larman, C. Applying UML and Patterns. An Introduction to Object-Oriented Analysis and Design and Iterative Development. Third Edition / C. Larman. – 2004, Prentice Hall PTR. – 736 p.

130. Левыкин, В.М. Подход к использованию паттернов проектирования при работе с требованиями к информационной системе / В.М. Левыкин,

М.В. Евланов, М.А. Керносов // Системний аналіз. Інформатика. Управління (САІУ-2013): матеріали IV Міжнародної науково-практичної конференції (м. Запоріжжя, 13-16 березня 2013 р.). – Запоріжжя: КПУ, 2013. –С. 150-152.

131. Месарович, М. Общая теория систем: математические основы / М. Месарович, Я.Такахара.: пер. с англ. – М.: Мир, 1978. – 312 с.

132. Лачинов, В.М. Информодинамика или Путь к Миру открытых систем / В.М. Лачинов, А.О. Поляков. – СПб.: Издательство СПбГТУ, 1999. – 122 с.

133. Поляков, А.О. Информодинамическая общность систем [Электронный ресурс] / А.О. Поляков // Сайт «Теория информации». – Режим доступа: http://inftech.webservis.ru/it/information/index.html. – Заголовок с экрана.

134. Левыкин, В.М. Параллельное проектирование информационного и программного комплексов информационной системы / В.М. Левыкин. М.В. Евланов, В.С. Сугробов // Радиотехника. – 2006. – Вып. 146. – С. 89-98.

135. Васильцова, Н.В. Разработка метамодели требований к информационной системе / Н.В. Васильцова. М.В. Евланов, И.Ю. Панферова // АСУ и приборы автоматики. – 2004. – Вып. 129. – С. 19-27.

136. Левыкин, В.М. Подход к формализации требований к информационной системе / В.М. Левыкин. М.В. Евланов, М.Ю. Хрипкова // Тез. докл. Междунар. конф. «Теория и техника передачи, приема и обработки информации». – Харьков: ХНУРЭ, 2003. – С. 242-243.

137. Lassila O. Frames or Objects, or Both? / O. Lassila // Workshop Notes from the Eight National Conference on Artificial Intelligence (AAAI-90): Object-Oriented Programming in AI, Boston (Massachusetts, U.S.A.), 1990. – 8 p.

138. Wu X. A Comparison of Objects with Frames and OODBs / Wu X. // Object Currents, 1996. – Vol 1.– No 1. – 9 p.

139. Минский, М. Фреймы для представления знаний / М. Минский. – М.: Энергия, 1979. –152 с.

140. Искусственный интеллект: в 3-х кн. Кн. 2. Модели и методы: Справочник / Под ред. Д.А. Поспелова – М.: Радио и связь, 1990. – 304 с.

141. Гаврилов, А.В. Системы искусственного интеллекта / А.В. Гаврилов. – Новосибирск: НГТУ, 2004. – 59 с.

142. Savitch, W. Java: An Introduction to Computer Science and Programming / W. Savitch: 2d ed. – Pearson: Prentice Hall, Inc, 2001 – 1039 p.

143. Deitel, H.M. C++ How to Program / H.M. Deitel, P.J. Deitel: 5th ed. – Pearson: Prentice Hall, Inc, 2005 – 1536 p.

144. Левыкин, В. М. Исследование и разработка фреймовой модели структуры документа / В. М. Левыкин, М. А. Керносов // Нові технології. – 2008. – № 1 (19). – С. 149–154.

145. Фленов, М.Е. Библия Delphi / М.Е. Фленов. – СПб.: БХВ-Петербург, 2004. – 880 с.

146. Євланов, М.В. Визначення лексикона візуального моделювання інформаційних систем / М.В. Євланов // Науковий вісник Інституту економіки та нових технологій „Нові технології”. –2004. – № 1-2 (4-5). – С. 204-208.

147. Booch G. Object-oriented Analysis and Design with Applications / G. Booch: 3d ed. – Reading: Addison-Wesley, 2007. – 693 p.

148. Левыкин, В. М. Информационная технология динамического формирования интерфейса пользователя системы управления электронным документооборотом / В.М. Левыкин, М. А. Керносов // Вісник ХНТУ. – 2008. – № 1 (30). – С. 182–188.

149. Новиков, А. Модель качества разработок CMM и ее поддержка линейкой продуктов Rational [Электронный ресурс] / А. Новиков // Сайт «Interface. Internet & Software company». – Режим доступа: http://www.interface.ru /fset.asp?Url=/rational/news/m010625307.htm. – Заголовок с экрана.

150. COCOMO II Model Definition Manual [Электронный ресурс] // Сайт «Center for Systems and Software Engineering». – Режим доступа: ftp://ftp.usc.edu/pub/soft_engineering/COCOMOII/cocomo99.0/modelman.pdf. – Заголовок с экрана.

151. Banker, R. An Empirical Test of Object-Based Output Measurement Metrics in a Computer Aided Software Engineering (CASE) Environment [Электронный ресурс] // R. Banker, R. Kaufmann and R. Kumar // Сайт «Social Science Research Network». – Режим доступа: http://papers.ssrn.com/sol3/papers.cfm?abstract_id=1289048. – Заголовок с экрана.

152. Banker, R. Evidence on Economies of Scale in Software Development [Электронный ресурс] / R. Banker, H. Chang and C. Kemerer // Сайт «CiteSeerXβ». – Режим доступа: http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=D9CBBA859982C4EA8CA13E2795DF2F85?doi=10.1.1.115.350&rep=rep1&type=pdf. – Заголовок с экрана.

153. Kaufmann, R. Modeling Estimation Expertise in Object-Based ICASE Environments / R. Kaufmann, R. Kumar // Stern School of Business Report, New York University, January 1993.

154. На старт! Внимание! И? [Электронный ресурс] // Сайт «ITCua». – Режим доступа: http://itc.ua/articles/na_start_vnimanie_i_21814/. – Заголовок с экрана.

Для заметок

Наукове видання

ЛЕВИКІН Віктор Макарович
ЄВЛАНОВ Максим Вікторович
КЕРНОСОВ Максим Андрійович

Паттерни проектування вимог до інформаційних систем:
моделювання й застосування
Монографія

План 201_ р., поз. __
Підп. до друку __.__.__. Формат 60х84 1/16.
Умов. друк. арк. Х,Х . Облік. вид. арк. Х,Х. Спосіб друку – ризографія.
Тираж 300 прим. Зам. 1-90 . Ціна договірна.

ХНУРЕ, 610166, Харків, просп. Леніна, 14

Віддруковано в навчально-науковому видавничо-поліграфічному центрі ХНУРЕ
61166 Харків, просп. Леніна, 14