УДК 519.713

В.С. СУГРОБОВ

МЕТОД РАЗРАБОТКИ ИНФОРМАЦИОННОЙ СИСТЕМЫ НА ОСНОВЕ ПРЕОБРАЗОВАНИЯ МОДЕЛЕЙ

Предлагается метод разработки ИС, основанный на преобразовании моделей. Реализация данного метода подразумевает внедрение в существующие инструментальные средства дополнительных возможностей для задания схем преобразований модели ИС и ее автоматической трансформации согласно указанным схемам преобразования. Последовательное применение ряда трансформаций должно, в конечном итоге, приводить к такой модели ИС, которая позволит сгенерировать готовые решения по заданному виду обеспечения ИС. Данный метод предлагается как эволюционный шаг на пути к созданию полноценных инструментальных средств, поддерживающих подход MDA.

1. Введение

Модель — это согласованный набор формальных элементов, описывающих некую сущность[1]. При проектировании ИС модели помогают в решении ряда задач, в том числе: в организации обмена информацией между людьми и компьютерами, проверке полноты и завершенности системы, генерации тестов, приблизительных оценок стоимости и эффективности, соответствии стандартам, преобразовании спецификации в реализацию.

Каждая модель описывает определенный круг тем. Мы можем построить модель ИС, которая будет описывать определенный аспект ИС, будь то безопасность или пользовательский интерфейс. Мы выбираем, какие элементы модель должна описывать, а какие можно исключить, хотя впоследствии, возможно, придется объединять несколько моделей. В данной статье под моделью ИС понимается совокупность множества моделей, описывающих различные аспекты проектируемой ИС. Примером таких моделей могут служить модель данных ИС и модель предметной области ИС [2], каждая из которых описывает определенную часть ИС и может быть включена в модель ИС.

Кроме того, мы можем представить модель в виде конструкций, которые существуют на некотором уровне языковой абстракции. Модель, описанная на языке Си, «абстрагируется» от реализации вызовов функций и вычисления выражений, оставляя все машинно-зависимые операции (например, распределение регистров) компилятору. Аналогичным образом модель, описанная на языке UML [3], проигнорирует организацию связей, возложив принятие соответствующих решений на компилятор модели или человека, занимающегося проектированием. Повышение уровня абстракции позволяет разрабатывать все более сложные системы. Это привело к тому, что структурные методологии и технологии (особенно при проектировании программного обеспечения ИС) сменились объектно-ориентированными. Следующим шагом на этом пути стало появление подходов к разработке ИС на базе моделей (model-driven development, MDD) [4]. Основой для модельно-ориентированной разработки стал MDA [5].

В основе MDA лежат понятия платформо-независимой и платформо-зависимой моделей (platform-independent and platform-specific models, PIMs and PSMs). В процессе разработки системы сначала создаётся PIM, содержащая бизнес-логику системы без конкретных деталей её реализации, которые относятся к какой-либо технологической платформе. Принципиальным является именно тот факт, что на этапе создания этой модели не принимается никаких решений по поводу её реализации, разрабатываемый программный продукт не привязывается к технологиям. На этом этапе в модель закладывается бизнес-логика, сценарии использования, функциональные требования и другая информация о взаимодействии системы с пользователем и о желаемом поведении ИС. При использовании MDA рекомендуется доводить платформо-независимую модель по достаточно высокой степени детализации, вплоть до использования высокоуровневого платформо-независимого языка программирования для описания функциональности и создания исполняемой модели. Однако следует отличать детали функциональности, описывающие поведение системы с точки зрения пользователя, от деталей её практической реализации: последние не должны присутствовать в платформо-независимой модели.

После того, как PIM в достаточной степени детализирована, выполняется переход к PSM. Эта модель описывает уже не только функциональность ИС, но и реализацию её обеспечений с использованием конкретной (выбранной для данного проекта) технологической платформы. Происходит дальнейшая детализация модели и добавление элементов и конструкций, специфичных для выбранной технологии реализации. После того, как модель достаточно разработана, выполняется генерация кода ПО ИС, затем производится доработка этого кода и его компиляция, так же как и в традиционных методиках разработки ПО ИС [6]. Основным языком для описания моделей при разработке ПО ИС с использованием подхода MDA является UML [3].

В последних версиях стандарта UML появились дополнения, которые делают этот язык более удобным для такого использования: язык действий (action semantics) [7] позволяет описывать функциональность ИС на уровне PIM, а профили (profiles) облегчают создание PSM.

Использование MDA позволяет получить ряд дополнительных преимуществ:

- повышает производительность труда разработчиков благодаря генерации кода ПО и артефактов разработки из модели ИС;
- повышает эффективность при многократном использовании, что объясняется тем, что отдача от средств, инвестированных в разработку трансформаций, происходит каждый раз при повторном использовании этих трансформаций;
- применение надежных и испытанных трансформаций повышает предсказуемость в разработке новых функций ИС и снижает проектный риск;
- облегчает переход на новую технологическую базу.

Генерация кода по модели UML достаточно хорошо разработана до появления подхода MDA и поддерживается практически всеми инструментальными средствами для работы с UML. Поэтому переход от PSM к коду особого интереса не представляет.

Автоматизированный же переход к PSM - это новая и недостаточно разработанная идея. Так как и PIM, и PSM - это модели, представленные на языке UML, то переход между ними, по сути, является трансформацией UML-модели по заданному описанию трансформации (содержащему формальным образом заданное описание перехода от UML-модели общего вида к конкретной технологии).

При использовании подхода MDA разработчик сталкивается с двумя проблемами. Во-первых, построение полной модели PIM является нетривиальной задачей. Это приводит к тому, что многие разработчики отказываются от моделирования на определенной стадии проектирования ИС и не строит полную PIM [1]. Во-вторых, современные инструментальные средства в недостаточной степени поддерживают подход MDA в плане автоматизированного перехода от PIM к PSM. Именно решению этих проблем и посвящена данная работа.

2. Подходы к трансформации моделей

Существует несколько способов описания и выполнения трансформаций моделей [8]. Простейший способ - это явное императивное описание процесса трансформации с использованием любого алгоритмического языка. При этом подходе в среду разработки на этапе её создания встраивается набор трансформаций, которые позднее могут быть задействованы пользователем. У этого подхода имеется ряд недостатков. Прежде всего, у пользователя отсутствует возможность добавлять новые описания трансформаций или изменять существующие; он вынужден использовать то, что сделано разработчиками инструмента. Кроме того, из-за отсутствия единого стандарта описания трансформаций разные среды разработки неминуемо будут выполнять трансформации по-разному даже для одной и той же технологической платформы, что может привести к возникновению случаев несовместимости и затруднит смену сред разработки. И наконец, подобный подход означает, что для каждой среды разработки придётся писать полный набор описаний трансформаций для всех популярных технологий.

Другой подход - использование уже разработанных механизмов трансформаций и преобразований из других областей информатики. В частности, можно представить модель в виде графа и использовать математический аппарат трансформации графов [9]. Главный недостаток такого подхода состоит в том, что в нем используется собственный понятийный аппарат, не имеющий отношения к UML-моделированию. Это значит, что от пользователей такой системы требуется знание не только UML-моделирования, но и теории графов и принципов их трансформации. Кроме того, поскольку UML-модель несёт семантическую

нагрузку, отличную от формального графа, правила трансформации, сформулированные для графа, будут трудны для понимания с точки зрения UML-модели: для понимания трансформации придётся мысленно совершать переход от графа к породившей его UML-модели, а для внесения изменений в описание трансформации - от UML к графу.

Ещё один вариант заключается в использовании методик трансформации XML-документов и стандарта XMI [10]. XMI (XML Metadata Interchange) - это стандарт, позволяющий представить UML модель в виде XML документа. Он предназначен главным образом для хранения UML-данных, а также любых других данных, метамодель которых задана с помощью MOF (Meta Object Facility) [5] и обмена ими между различными инструментами и средами разработки. MOF - это стандарт описания метамоделей, с помощью которого в частности можно описать структуру и общий вид UML-модели. Для XML существует несколько хорошо развитых методик трансформации, в частности XSLT [11] и XQuery [12]. Для трансформации UML-модели можно преобразовать её в XMI-представление, выполнить трансформацию средствами работы с XML, и затем преобразовать результат обратно в UML. Но XMI разрабатывался, прежде всего, как стандарт хранения и обмена UML-данными, он сложен для чтения и понимания пользователем. Также очень сложно понять функционирование трансформации, описывающей преобразование XML-документа, с точки зрения UML-модели, которой соответствует этот документ. Из-за того, что трансформация описывается в терминах XML, а не UML, большая часть описания трансформации оказывается направленной на то, чтобы в результате получить XML-документ, соответствующий стандарту XMI, а не на собственно описание трансформации UML.

Язык моделирования UML считается универсальным и содержит средства описания трансформаций. В частности, стандарт CWM (Common Warehouse Metamodel) позволяет описывать трансформации и преобразования [13]. Идея использовать UML для описания трансформаций UML-моделей подобно тому, как на UML описывается синтаксис UML, выглядит заманчиво, но трудно реализуема на практике. CWM даёт возможность только описывать сам факт того, что между определёнными элементами модели существует отображение, но не содержит развитых средств для задания трансформаций в общем виде (декларативно или императивно). Ещё один стандарт из семейства UML - QVT (Query, View, Transformation) - представляет значительно больший интерес с точки зрения его применения в MDA. Но, к сожалению, на данный момент этот стандарт находится на ранних этапах разработки. Кроме того, вызывает опасение тот факт, что в рамках этого стандарта предполагается решить сразу несколько общих задач, и то, что QVT ориентирован прежде всего не на практическое применение, а на развитие концепции метаметамоделирования в рамках MOF (Meta Object Facility). Существует вероятность того, что из-за подобной направленности на теорию и на решение задач в общем виде этот стандарт будет неудобен для использования на практике в задачах, специфичных для MDA, хотя ни в чём нельзя быть уверенным, пока стандарт ещё не принят хотя бы в ранней версии.

Еще одним вариантом является разработка языка и средства трансформации моделей, предназначенного именно для применения в MDA. Возможно, такое средство трансформации окажется более удобным и эффективным в данной узкой области, чем адаптация более универсального стандарта. Преимуществом такого языка является его узкая специализация, направленная на решение задач MDA. Недостаток – отсутствие единого стандарта для такого языка, что делает затруднительным его внедрение во множество инструментальных средств. Также от разработчика потребуется изучение дополнительного языка трансформации[6].

Все рассмотренные подходы к трансформации моделей имеют недостатки. В рамках прелагаемого метода будет рассмотрен подход к трансформации моделей, представляющий собой синтез некоторых из рассмотренных в данном разделе подходов. Такой синтез позволяет устранить ряд недостатков указанных подходов и решить проблему использования современных инструментальных средств при проектировании ИС с использованием подхода MDA и преобразований модели.

3. Метод разработки ИС на основе преобразования моделей

Каждая организация, занимающаяся разработкой ИС, имеет набор формальных и неформальных знаний в виде шаблонов проектирования, унифицированных интерфейсов пользователя, методов доступа к базам данных, средств организации взаимодействия между

прикладными программами, различных техник и методик создания элементов различных обеспечений ИС и т.п. Данный набор знаний будем далее называть каталогом решений. Каталог решений используют для быстрого решения различных проблем проектирования ИС, которые возникали ранее и, в большинстве случаев, применяются при разработке нескольких ИС. Каталог решений может использоваться для выявления типовых проблем проектирования и предоставлять варианты их решения. Описание каталога решений в виде схем преобразования моделей ИС, реализация трансформаций моделей ИС согласно решениям из каталога решений в инструментальных средствах являются фундаментом метода разработки ИС на основе преобразования моделей.

В то же время организация-разработчик ИС хорошо знакома с предметной областью (ПрО), для которой разрабатывается ИС. Большинство организаций-разработчиков обладают достаточной информацией, чтобы построить детальную модель ПрО. Некоторые организации-разработчики уже имеют модели ПрО (а иногда и модели ИС). Использование метода разработки ИС на основе преобразования моделей подтолкнет организации-разработчики к созданию (на основе имеющихся моделей ПрО и ИС) более детальных моделей ИС, которые далее смогут быть использованы как основа для построения PIM и перехода к проектированию ИС с применением подхода MDA.

Выше говорилось о возможности реализации автоматического преобразования моделей ИС с помощью инструментальных средств. Реализовать преобразования моделей наиболее просто, встроив распространенные преобразования непосредственно в инструментальное средство. Такой подход имеет ряд преимуществ, к которым можно отнести удобство работы, надежность решений, простота использования. Однако он сталкивается с непреодолимыми препятствиями, если возникает необходимость в специфических преобразованиях моделей. Такие специфически преобразования в большинстве случаев потребуются при использовании каталога решений организации-проектировщика при создании ИС. В этом случае целесообразно использовать другой способ (возможно в комбинации с первым). Суть этого способа заключается в предоставлении пользователям CASE-средств возможности оперировать репозиторием данного средства через механизм расширений. В этом случае организация-разработчик сможет самостоятельно (учитывая, что организации-разработчики в большинстве случаев обладают квалифицированным персоналом) реализовать свой каталог решений в виде расширений CASE-средства. Такой подход позволит реализовать любые требуемые преобразования моделей и использовать расширения, созданные другими организациями-разработчиками, что, в свою очередь, сократит затраты при проектировании ИС. При этом наиболее востребованные расширения, а значит, наиболее удачные решения из каталога решений в плане разработки ИС будут собираться в пакеты расширений, которые в дальнейшем будут использоваться в рамках MDA для преобразования PIM в PSM. Интегрирующим началом таких инструментальных средств и расширений должны стать XMI, CWM и MOF.

С помощью спецификации CWM можно задать структуру преобразования модели, ее входные и выходные элементы, оставив реализацию самого преобразования на усмотрение пользователя.

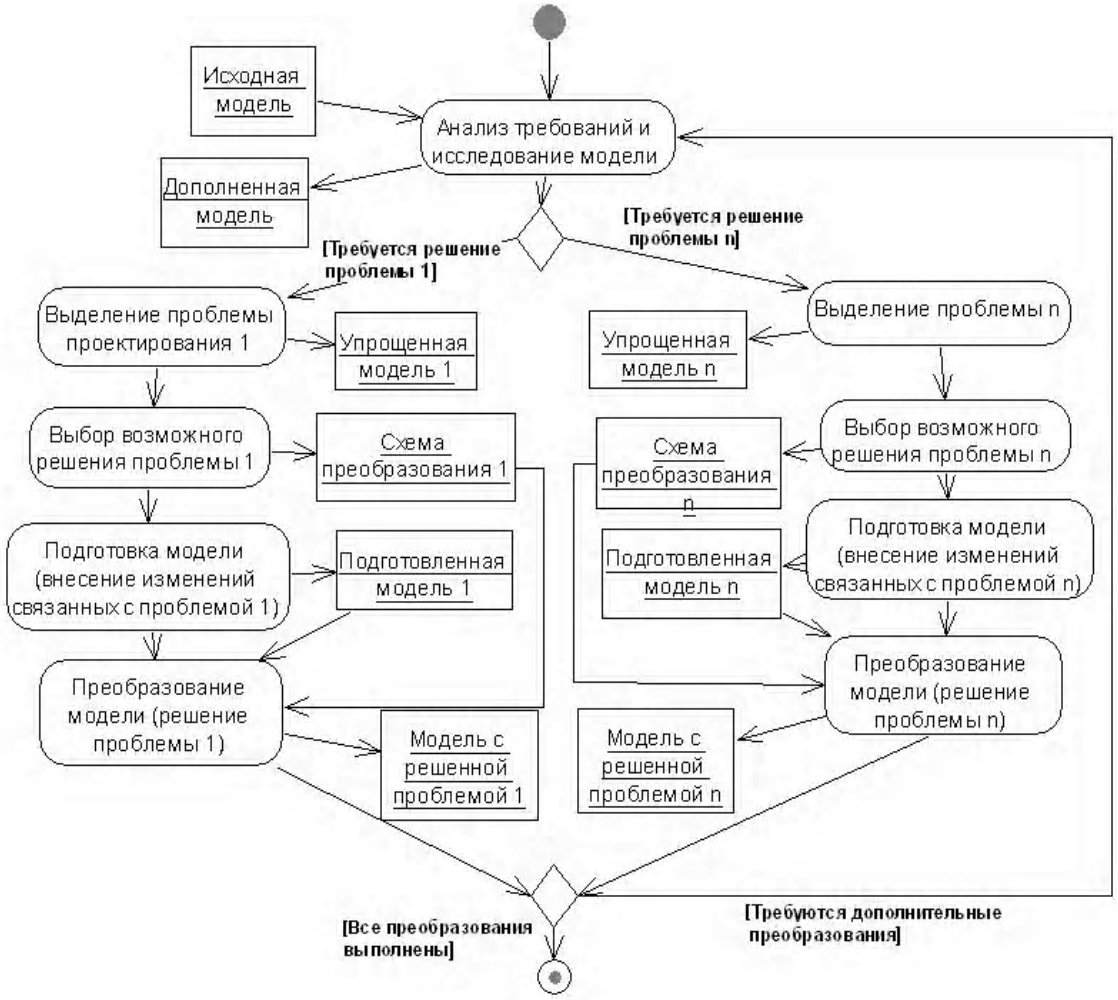
Таким образом, развитием инструментальных средств является поэтапное наращивание их возможностей за счет расширений, в которых будут реализованы различные преобразования моделей. При этом преобразование моделей будет происходить не по схеме PIM $\rightarrow$ PSM, а по более простой схеме. Входной моделью будет выступать модель, отражающая некоторый простой аспект ИС, и преобразование будет направлено на решение одной, часто встречающейся проблемы разработки ИС. Примером такой часто встречающейся проблемы является взаимодействие слоя ПрО и слоя данных [2].

Такой подход дает возможность получать экономический эффект сразу после реализации данного преобразования. Реализация множества таких простых преобразований (накопленных со временем), в конечном итоге, позволит говорить о реализации преобразования PIM $\rightarrow$ PSM. Одновременно с этим процессом будет происходить уточнение моделей разрабатываемых ИС, на основании которых в дальнейшем будет разработана PIM система. Оптимальным вариантом является разработка единых PIM для однотипных ИС, которые будут доступны всему сообществу разработчиков.

Таким образом, метод разработки ИС на основе преобразования моделей можно представить в виде итеративной последовательности следующих этапов:

1) анализ требований и исследование модели ИС;
2) выделение проблемы проектирования (выделение модели некоторого аспекта ИС);
3) выбор возможного варианта решения проблемы;
4) подготовка модели для последующего преобразования;
5) преобразование модели.

Структурная схема метода представлена на рисунке.



Структурная схема метода разработки ИС на основе последовательного преобразования моделей

На первом этапе анализа требований и исследования модели ИС разработчик определяет, какая проблема на данной итерации требует первоочередного решения и добавляет в исходную модель детали, выявленные в результате анализа требований. Входными данными для этого этапа является исходная модель ИС. Степень детализации исходной модели должна быть достаточна для анализа требования и дальнейших ее преобразований. Результатом выполнения первого этапа является дополненная модель ИС и выбранная проблема проектирования, решением которой будет заниматься разработчик ИС.

На втором этапе выделения проблемы проектирования разработчик исключает из рассмотрения все детали, которые не касаются решаемой проблемы проектирования, тем самым упрощая модель ИС (выделяя модель некоторого аспекта ИС) и фокусируя внимание на решении данной проблемы проектирования. Входными данными для этапа является дополненная модель ИС. Результатом выполнения второго этапа является упрощенная модель ИС, отражающая только необходимую для решения данной проблемы информацию.

На третьем этапе выбора возможного варианта решения проблемы проектирования разработчик выбирает одно из уже существующих решений данной проблемы или разрабатывает собственное. Входными данными для этапа является упрощенная модель ИС. Выбранное решение, как результат выполнения этапа, представляется в виде схемы преобразования модели ИС. Может быть использована уже существующая схема преобразования (как созданная разработчиком ИС ранее, так и созданная сторонними разработчиками). Выбор того или иного решения проблемы проектирования может сильно зависеть от того, существует ли для данного решения схема преобразования модели ИС или ее придется создавать заново. Использование существующей схемы преобразования значительно упрощает применение метода разработки ИС на основе преобразования моделей и снижает затраты на разработку ИС.

На четвертом этапе подготовки модели разработчик вносит дополнительную информацию в модель ИС, которая позволит осуществить автоматическую трансформацию модели в модель, которая будет содержать решения данной проблемы проектирования. Входной информацией для этапа является схема преобразования и упрощенная модель ИС. Результатом выполнения этапа является подготовленная модель ИС, которая содержит дополнительную информацию, необходимую для дальнейшего преобразования.

Этап преобразования модели предполагает трансформацию подготовленной модели ИС в модель, содержащую решение данной проблемы проектирования. После трансформации модель, содержащая решение проблемы проектирования, синхронизируется с исходной моделью. Использование метода разработки ИС на основе преобразования моделей будет эффективным только в случае полной автоматизации данного этапа в инструментальных средствах с применением подхода, описанного выше. Входными данными для этапа является подготовленная модель ИС. Результатом выполнения этапа является модель ИС с решенной проблемой проектирования. Далее полученная модель ИС может быть использована как исходная для решения другой проблемы проектирования или для генерации решений по различным видам обеспечений ИС.

4. Выводы и перспективы развития

Метод разработки ИС на основе преобразования моделей базируется на подходе MDA. Данный подход поддерживается рядом стандартов и технологий, на практике доказавших свою полезность. Концептуальной основой появления MDA стали спецификации OMA, ORB, CORBA. Перевести замысел в практическую плоскость позволили технологии объектно-ориентированного программирования, стандарт CWM, языки UML, XML, MOF. Работами по созданию новой архитектуры программирования занялся консорциум Object Management Group. По мнению создателей, архитектура MDA является новым витком эволюции технологий программирования, так как описывает процесс разработки в целом. Новизна MDA, в отличие от существующих стандартов, заключается в том, что описание процесса разработки в ней выполнено с использованием современных средств представления и позволяет автоматизировать создание ИС.

Подход к проектированию ИС, предлагаемый МДА, дает следующие преимущества:

- независимость модели от средств разработки обеспечивает возможность реализации на любой программной платформе;
- сокращение работ при переносе системы на другую платформу;
- экономия ресурсов при создании систем для нескольких платформ одновременно;
- возможность тестирования системы на ранних этапах проектирования;
- автоматизация процесса проектирования ИС, обеспечивающая автоматическое создание типовых частей ИС, таких как пользовательский интерфейс, программная реализация типовых операций, организация доступа к данным и др;
- упрощает проектирование систем, использующих несколько различных платформ, предлагает описания межплатформенного взаимодействия;
- позволяет использовать несколько репозиториев, применяя спецификации XMI, CWM и MOF;
- дает предпосылки для разработки исполняемых моделей.

Однако на сегодняшний день существует ряд существенных недостатков, связанных с разработкой ИС с использованием MDA. К ним можно отнести следующие:

- отсутствие полноценной поддержки подхода MDA в современных инструментальных средствах, что делает невозможным в настоящее время применение подхода MDA при разработке ИС;

- OMG, в спецификации MDA, предлагает подход к созданию ИС с использованием исполняемых моделей, при этом не предоставляет технологий и инструментальных средств, реализующих данный подход.

Предлагаемый метод разработки ИС на основе преобразования моделей использует идеи подхода MDA, но не является его реализацией. Это его недостаток, так как теряется ряд преимуществ, предполагаемых подходом MDA. Однако данный метод позволяет избежать некоторых проблем при проектировании ИС, которые возникают при использовании подхода MDA. К таким проблемам можно отнести разработку детальных моделей ИС. Предлагаемый метод, в отличие от MDA, не требует детальной модели ИС, для его работы достаточно сравнительно простых моделей некоторых аспектов ИС. Еще одним преимуществом метода разработки ИС на основе преобразования моделей является то, что он позволяет разработчикам ИС самостоятельно реализовывать в инструментальных средствах необходимые им преобразования моделей (при условии поддержки инструментальными средствами механизма расширений). Такой подход к разработке ИС является эволюционным путем к реализации подхода MDA. Метод разработки ИС на основе преобразования моделей более гибкий и простой в реализации и использовании, чем подход MDA. Разработка множества методов, базирующихся на методе разработки ИС на основе преобразования моделей, позволит эволюционным путем перейти к использованию подхода MDA при разработке ИС.

Список литературы: 1. *Mellor, S.* Model-Driven Development [Text] / Stephen Mellor, Anthony Clark, Takao Futagami // IEEE Software. IEEE Computer Society. 2003. 2003 September/October 2003. 2. *Фаулер, М.* Архитектура корпоративных программных приложений [Текст]/ М. Фаулер. М.: «Вильямс», 2004. 544 с. 3. *Unified* Modeling Language Specification [Электронный ресурс]: спецификация. Object Management Group, 2005. Режим доступа: http://doc.omg.org/formal/05-07-04. Загл. с экрана. 4. *Bran Selic*, The Pragmatics of Model-Driven Development [Text] // IEEE Software. IEEE Computer Society. 2003. 2003 September/ October 2003. С. 26. 5. *Miller, J.* MDA Guide Version 1.0 [Электронный ресурс]: инструкция /Joaquin Miller, Jishnu Mukerji(eds.); Object Management Group, 2003. Режим доступа: http://www.omg.org/docs/omg/03-06-01.pdf. Загл. с экрана. 6. *Кузнецов, М.Б.* Трансформация UML-моделей и ее применение в технологии MDA [Электронный ресурс] / М.Б. Кузнецов //Препринт Института Системного Программирования РАН. Режим доступа: http://www.citforum.ru/SE/project/uml_mda/. 7. *Varro, D.* ModelUML Action Semantics For Model Transformation Systems [Text] / Daniel Varro, Andras Pataricza // Periodica Polytechnica. 2003. № 3-4. 8. *Czarnecki, K.* ModelUML Classification of Model Transformation Approaches [Text] / Krzysztof Czarnecki, Simon Helsen// University of Waterloo. 2003. 9. *Agrawal*. ModelUML Graph Transformations on Domain-Specific Models [Text] / Agrawal, G. Karsai and F. Shi.// Journal on Software and Systems Modeling. 2003. 10. *Kovse, J.* Generic XMI-Based UML Model Transformations [Text] / Jernej Kovse, Theo Hдrder // ZDNet UK Whitepapers. 2002. 11. *XSL* Transformations (XSLT) v1.0 [Электронный ресурс]: W3C Recommendation, Nov. 1999. Режим доступа: http://www.w3.org/TR/xslt.html . Загл. с экрана. 12. *Chamberlin, D.* XQuery: An XML query language [Text] // IBM systems journal. 2002. № 4. 13. *Common* Warehouse Metamodel (CWM) Specification [Электронный ресурс]: OMG Documents. Feb. 2001. Режим доступа: http://www.omg.org/cgi-bin/apps/doc?formal/03-03-02.pdf. Загл. с экрана.

Поступила в редколлегию 17.03.2008

Сугробов Владимир Сергеевич, аспирант кафедры ИУС ХНУРЭ. Научные интересы: моделирование при проектировании ИС. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 80663837722.