УДК 519.686.4

ИСПОЛЬЗОВАНИЕ МЕТОДА УЧЕБНЫХ ПРОЕКТОВ НА БАЗЕ ТЕХНОЛОГИИ СОЦИАЛЬНЫХ СЕТЕЙ

О.Н. Купин[1], Н.С. Лесная[2], И.Н. Келеберда[3]

[1]ХНУРЭ, г. Харьков, Украина, devilforce@yandex.ru
[2]ХНУРЭ, г. Харьков, Украина
[3]ХНУРЭ, г. Харьков, Украина, i.keleberda@gmail.com

В ходе написания данной статьи был произведен анализ текущей ситуации в области учебных и научных социальных сетей, исследована структура учебного проекта, возможность адаптации его в области обучения информационным технологиям. Произведено описание архитектуры учебной социальной сети и определены требования для использования в данной сети метода учебных проектов. Построена модель учебного проекта в рамках концепции учебных социальных сетей.

PROBE, УЧЕБНЫЙ ПРОЕКТ, МЕТРИКИ, ЧАСТЬ, ОБЪЕМ ПРОГРАММНОГО ПРОДУКТА

Введение

Использование в системе образования дистанционных курсов привело педагогов к необходимости использования сетевых технологий в учебном процессе. Доступность Интернета обусловила целесообразность его использования в учебных и научных целях, а развитие социальных сетей и их большая популярность в последнее время приводит к необходимости спроецировать их в научно-педагогическое пространство. Одним из направлений использования социальных сетей является построение модели учебных и научных проектов в системе социальных сетей. Это очень перспективное направление в свете реалий современного обучения.

Анализ психолого-педагогической литературы показывает, что проектная деятельность и педагогов, и учащихся находится в процессе становления, обобщения эмпирических фактов и результатов исследования. Отсутствие необходимых знаний, умений, психологических обоснований проектирования – одна из серьезнейших проблем современного образования как учащихся, так и педагогов. Известно, что знание, оторванное от понимания, временами не имеющее ничего общего с опытом учащегося, практически не используется. В такой ситуации метод проектов, как один из способов, позволяет педагогу решить сложную задачу смены традиционного образования с безликой формы в личностную культуру учащегося [2].

Таким образом, использование метода проектов в учебном процессе высшей школы является целесообразным. Несмотря на то, что технология социальных сетей появилась относительно недавно, она имеет сейчас очень широкую аудиторию и представляет собой альтернативную модель общения, обмена информацией, а главное – систематизацию необходимой информации и легкого доступа к ней. Анализируя эти факторы, можно сделать вывод, что социальная сеть является наилучшим решением, которое необходимо взять как базовое для реализации инструментов, и разработать модель метода учебных проектов в интерактивном пространстве.

1. Анализ предметной области и постановка задачи

В данной статье исследуется метод учебных проектов в педагогической деятельности на примере построения модели создания программных продуктов в сфере обучения информационным технологиям.

Учебный проект с точки зрения студента – это деятельность, позволяющая проявить себя, попробовать свои силы, приложить свои знания, принести пользу и показать публично достигнутый результат; это деятельность, направленная на решение интересной проблемы, сформулированной самими учащимися в виде цели и задачи, когда результат этой деятельности – найденный способ решения проблемы – носит практический характер, имеет важное прикладное значение и, что весьма важно, интересен и значим для самих разработчиков.

2. Построение модели учебного проекта создания программного продукта

Введем пять этапов разработки учебного проекта, смысл каждого определяется его названием и комплексом решаемых задач: предварительный этап, этап начала взаимодействия студентов, этап анализа предметной области и написания документации, этап кодирования и тестирования программного продукта, этап выпуска и презентации.

2.1 Предварительный этап

На предварительном этапе преподаватель создает или подбирает из существующих в системе пользователей (студентов) группу для осуществления проекта. Создает сущность «проект» в системе и регистрирует пользователей в этой сущности. После чего настраивает возможности (индивидуально или всей группе) работы пользователей в проекте. Важным на этом этапе также является определение преподавателем сроков начала и окончания проекта. Дата и время начала проекта открывает для студентов, определенных на конкретный проект, доступ ко всем ресурсам, хранилищу данных, чатам, календарю, а дата окончания проекта – закрывает

этот доступ. Преподаватель также назначает время вводной дискуссии в проект и финальной дискуссии по окончанию проекта.

2.2 Этап начала взаимодействия студентов

Данный этап начинается с момента старта проекта, дату которого определил преподаватель на предыдущем этапе. После старта проекта за короткий промежуток времени преподаватель проводит вводную дискуссию со студентами, во время которой определяет цели проекта, результат для конечного пользователя программного продукта, дает советы по организации работы в команде, специфике предметной области, предполагаемым источникам информации. Итогом данной дискуссии является документ с общими характеристиками программного продукта, который подготавливает преподаватель заранее и который представляет собой техническое задание.

Этот документ представляет собой обзор того продукта, который заказчик (в случае учебного проекта – это преподаватель) хочет получить от группы разработчиков. В рамках этого документа описываются цели, поставленные заказчиком, определяется список терминов (глоссарий), определяются основные проблемы, которые необходимо решить в ходе выполнения проекта, описываются пользователи, которые будут работать с этой системой после введения ее в эксплуатацию и их потребности. Определяются также возможности и ограничения изготавливаемого продукта, показатели качества, требования к документации и другие требования к системе.

После данной дискуссии студентами подготавливается график дискуссий, в рамках которых должны решиться следующие задачи: разбивка проекта на части, количество которых равно количеству человек в команде, назначение каждой части на участника команды, также определяются временные рамки для каждого из последующих этапов, согласовываются варианты коммуникации и других организационных вопросов.

2.3 Этап анализа предметной области и написания документации

После того, как система разбита на отдельные логические части и каждая часть закреплена за каждым отдельным студентом, начинается этап уже непосредственно разработки программного продукта. Первым этапом разработки является написание документации по программному продукту. В связи с тем, что за каждым закреплена отдельная часть с отдельным функционалом, после написания и согласования общей части документации происходит независимое написание документации по каждой из частей проекта. С периодичностью, определенной на предыдущем этапе, проходят дискуссии между членами команды с текущими отчетами о проделанной работе с момента последней дискуссии и с планами о проделываемой работе с момента окончания дискуссии и до начала следующей. Во время данных дискуссий происходит обмен опытом, полученной новой информацией, обсуждаются трудности в том или ином этапе разработки. Данные дискуссии начинают проводиться с данного этапа разработки и заканчиваются уже на последнем этапе, во время сдачи проекта.

Общим для всех частей системы и для дальнейшей работы с ней необходимо написать общие части документации для всей системы: необходимо определить список акторов в системе и их взаимодействие. Актор – это внешний фактор и его действия несут недетерминированный характер. В роли актора может выступать и программная система, если она инициирует выполнение некоторых работ, удовлетворяющих поставленной цели системы. Актор, как абстракция внешнего объекта, может быть человеком или внешней системой. В модели системы актор может быть представлен классом, а пользователь – экземпляром класса. Если актором является другая программная система, то он будет представлять ее интересы. При этом одно лицо может быть экземпляром нескольких акторов.

Если актор находится вне системы, то он взаимодействует с ней через сценарий, который инициализирует последовательность операций для выполнения системы. Когда пользователь, как экземпляр актора, вводит определенный символ для старта экземпляра соответствующего сценария, это приводит к выполнению ряда действий в системе, которые заканчиваются тогда, когда экземпляр сценария находится или в состоянии ожидания очередного входного символа или завершения сценария. Данная информация базируется на документе технического задания, предоставляемом преподавателем.

Затем для каждой из частей определяется реестр вариантов использования.

Варианты использования это – описание последовательности действий, которые может осуществлять система в ответ на внешние воздействия пользователей или других программных систем. Варианты использования отражают функциональность системы с точки зрения получения значимого результата для пользователя, поэтому они точнее позволяют ранжировать функции по значимости получаемого результата.

Далее каждый определяет функциональные требования и бизнес-требования для своей части в целом. Функциональные требования к системе определяют действия системы, которые она должна выполнять. Функциональные требования реализуются через функции системы. Под функцией автоматизированной системы подразумевается совокупность действий автоматизированной системы, направленная на достижение определенной цели

или аспект определенного поведения системы, а под задачей – функция или часть функции системы, представляющая собой формализованную совокупность автоматических действий, выполнение которых приводит к результату заданного вида. Бизнес-требования описывают, каким образом разрабатываемая система связана с достижением бизнес-целей организации и что она должна для этого делать. Можно сказать, что бизнес-требования являются своего рода задачами, которые должна решать автоматизированная система для достижения целей своего создания.

Последним на данном этапе разработки документации является определение интерфейса пользователя. Здесь описывается, как должен взаимодействовать конечный пользователь с модулями системы.

После определения требований к системе разработка проекта переходит в стадию планирования. Известно, что чем точнее будут определены требования, тем более точным будет произведено планирование. Основной единицей планирования являются следующие оценки: времени, необходимого количества рабочих часов разработчиков, инженеров по качеству, аналитиков и менеджеров, необходимых для успешного завершения проекта. Изначально оценка затрачиваемого времени является неопределенной единицей, так как неясно, насколько будет велик конечный продукт. Чем раньше будет произведена оценка затрачиваемого времени, тем менее точной она будет. Преимущество в использовании определенных методов оценки затрачиваемого времени в том, что команда имеет механизм, который посредством приобретения опыта может улучшать процесс реализации проектов: имеется основа для сбора информации по данному методу, возможность использовать статистику оценки времени для более точного планирования в последующих проектах. В модели учебного проекта было решено использовать метод PROxy Based Estimating (метод оценивания времени, основанный на частичных объектах), сокращенно PROBE.

Метод оценивания времени, основанный на частичных объектах, базируется на том, что каждый студент создает компонент, опираясь при этом на те, которые он создавал ранее. Такие компоненты называются частями. Части – это единица программной реализации, представляющая файл, метод, объект, скрипт, страницу.

Критерии для точного определения части:

– размер части должен зависеть от необходимых усилий для завершения проекта;

– содержание части должно быть конечным и иметь определенную величину;

– часть должна быть легко отображаемой визуально и интуитивно понятна;

– часть должна быть легко изменяема;

– часть должна быть чувствительна ко всем изменениям в процессе реализации.

Для того чтобы часть была полезной, она должна быть тесно связана с требованиями к разработке программного продукта. Объективно оценив размер части, можно оценить размер необходимой работы по компоненту, который эта часть отображает. Определить эффективность части можно на основе истории создания разработанных ранее подобных частей. Также очень важно, чтобы язык программирования, стили и категория разрабатываемого приложения, которые будут использоваться в проекте, были определены до этапа расчета оценки затрачиваемого времени.

Многие потенциальные типы частей могут удовлетворять поставленным выше ограничениям. Программный метод является очевидным примером для использования его как части. Действительно, очень удобно, имея детальный дизайн, принять метод как часть и определять затрачиваемое время на написание метода (части). Также возможно использовать в виде части программные окна, вебстраницы, объекты, скрипты, разделы документации и так далее.

Принципы объектно-ориентированного проектирования позволяют предположить, что объекты будут как нельзя лучше подходить на роль частей. Объекты – это физические сущности, размер которых может быть определен программно. Удобство использования объектов, как частей, связанно с тем, что количество строк кода (которые являются важной метрикой) соотносится с часами, затраченными разработчиком, так же как и объекты, которые в свою очередь состоят из строк кода. Таким образом, объекты, являющиеся в конечном итоге физическими величинами, очень удобно использовать в качестве частей. При таком выборе необходимо учитывать наличие опыта написания кода именно на том языке, который является требованием планируемого объекта.

По результатам оценки времени, затрачиваемого на проект, необходимо создать концептуальный дизайн системы. Этот дизайн предусматривает разбиение системы на объекты вплоть до названия методов данной системы.

Следующей не менее важной частью разработки документации является написание детального дизайна системы, декомпозированного вплоть до самых нижних объектов системы. Этот документ является завершающим в цепочке спецификаций и позволяет полностью сосредоточиться на стадии кодирования системы. Спецификация описывает низкоуровневую организацию продукта. Здесь для каждого модуля разработчик должен определить все требования, включая передаваемые параметры, глобальные структуры и переменные, вызыва-

емые подпрограммы и так далее. Эта информация важна для программистов, которые параллельно реализуют различные модули, взаимодействующие друг с другом. Хорошо выполненная техническая спецификация снимает все проблемы, возникающие при объединении отдельных модулей в единое целое. Вторая часть низкоуровневого проектирования – создание псевдокода – является трудной, но очень важной частью процесса. Программисты не любят писать псевдокод, поскольку им кажется, что непосредственное кодирование осуществить быстрее. Часто это действительно так. Для маленьких процедур и модулей код размером в несколько строк написать проще, при этом не надо тратить время на представление одних и тех же действий дважды. Однако для крупных проектов с большой командой усилия, затраченные на эту работу, с лихвой окупятся на этапе сопровождения продукта, делая его более понятным и модифицируемым с меньшими затратами.

Имея концептуальный дизайн, название каждого объекта и категорию, далее необходимо найти объекты в базе данных предыдущих проектов, которые являются аналогичными объектам создаваемой системы.

В каждой из частей рекомендуется использовать код, написанный ранее. Тогда при оценке времени данной части можно ссылаться на количество затраченного времени на подобную часть из истории.

Если студент находит созданные ранее объекты или процедуры, которые будут отвечать концептуальному дизайну, он может использовать их повторно. Для любых объектов, при их использовании повторно, необходимо внести их имена и размер в строку повторно использованных объектов. Данный метод оценки времени подразумевает два типа повторно используемых объектов: повторно используемые объекты из библиотеки объектов и вновь создаваемые вспомогательные объекты, которые пересекаются с данной частью своим функционалом. Данные объекты определяются как повторно используемые [1].

Оцененные новые и измененные строки кода рассчитываются – $N = B0 + B1 * E$. Метод линейной регрессии производит расчет статистических параметров. Согласно этому методу определяется 2 параметра β_0 и β_1, используемые в расчетах размера программного продукта.

Формула линейной регрессии, используемая для расчетов размера программного обеспечения, показана ниже. Параметры β_0 и β_1 рассчитываются с учетом статистических данных по предыдущим проектам:

$$\beta_1 = \frac{\left(\sum_{i=1}^{n} x_i y_i\right) - \left(n x_{avg} y_{avg}\right)}{\left(\sum_{i=1}^{n} x_i^2\right) - \left(n x_{avg}^2\right)}, \qquad (1)$$

$$\beta_0 = y_{avg} - \beta_1 x_{avg}. \qquad (2)$$

Таким образом, размер программного обеспечения рассчитывается следующим образом:

$$PS = \beta_{0size} + \beta_{1size} * (E). \qquad (3)$$

Затраченное время:

$$DT = \beta_{0time} + \beta_{1time} * (E). \qquad (4)$$

Вспомогательные и регрессионные параметры определяются для того, чтобы в конечном итоге при подсчете объема программы учесть опыт предыдущих разработок.

Прогнозированный интервал представляет собой допустимые рамки оцененного объема или времени. Он не является обязательным для попадания, а всего лишь предположением. Широкий диапазон статистических данных будет иметь более широкий прогнозированный интервал. Формула расчета прогнозированного интервала попадания показана ниже, где n – количество данных точек; стандартное отклонение от регрессионной линии; t – распределенное значение вероятности p (70%); k – это оцененный размер, avg – среднее значение данных.

$$\text{Range} = t(p,n)\sigma \sqrt{1 + \frac{1}{n} + \frac{\left(x_k - x_{avg}\right)^2}{\sum_{i=1}^{n}\left(x_i - x_{avg}\right)^2}}. \qquad (5)$$

Стандартное отклонение измеряется разницей данных вокруг регрессионной линии. Широко разбросанные данные будут иметь более высокое стандартное отклонение, чем тесно сгруппированные. Ниже приведена формула расчета стандартного отклонения, которое равняется квадратному корню из дисперсии:

$$\text{Variance} = \sigma^2 = \frac{1}{n-2}\sum_{i=1}^{n}\left(y_i - \beta_0 - \beta_1 x_i\right)^2. \qquad (6)$$

Значение прогнозированного интервала предназначается для определения качества оценки с целью последующего ее использования в других частях.

Оцененное количество строк кода рассчитывается таким образом – $E = BA + NO + M$, где BA – суммарные базовые добавления частей, NO – суммарные новые объекты, M – количество строк, которые необходимо изменить для использования метода PROBE, что напрямую зависит от качества данных, которые используются. Если статистические данные имеют большой разброс в значениях, то прогнозированный интервал окажется достаточно большим.

Рассчитывается верхняя и нижняя граница прогнозированного интервала следующим образом: верхняя – $UPI = P +$ Range (где P – это оцененный размер части, а $Range$ – прогнозированный интервал), нижняя – $LPI = P -$ Range (или 0).

Задача разработки спецификаций проекта считается выполненной, если пользователь четко сформировал свои ожидания к результирующему продукту, а программист способен однозначно реализовать эти ожидания в продукте без любых других знаний о проекте, руководствуясь только спецификациями.

2.4 Этап кодирования и тестирования программного продукта

На этом этапе работа над документацией считается завершенной, и группа студентов приступает к реализации задокументированной функциональности. Отведенное время на этот этап разбивается на более мелкие итерации, во время которых производится планирование поэтапной реализации каждой части каждым студентом. После чего наступает сама реализация всех частей студентами.

По окончании реализации части кодирования и выпуска первой версии программы для бета-тестирования студенты начинают тестировать части друг друга. Причем очень важно, чтобы каждый студент тестировал не свою часть, а часть коллеги по команде. Во время тестирования и нахождения дефектов каждый дефект заносится в систему фиксации дефектов. После занесения дефекта студент, в чьей части он был найден, получает уведомление о том, что в его части был найден дефект. После чего происходит процедура исправления и последующей перепроверки неисправности. Таким же образом перед проведением тестирования студент пишет тест кейсы по своей части.

В период тестирования программного продукта очень важным аспектом является понятие качества и производительности кода. Качеством программного обеспечения называется характеристика программного обеспечения как степени его соответствия требованиям. Конечным определением же является удовлетворение потребностей пользователей. Причем потребностей, а не предпочтений, что является уже второстепенным фактором. Существует следующая иерархия потребностей пользователей: выполнение обязательных заданий, удовлетворение требований производительности, удобность в использовании, экономичность, современность и надежность.

Для того чтобы быть полезным, программное обеспечение должно быстро устанавливаться и легко запускаться, последовательно и правильно обрабатывая допустимые и недопустимые действия пользователя и не содержать критических ошибок. Ошибки, критически не важные для пользователя, являются таковыми, пока они не влияют на ключевые операции, не вызывают неудобства, не вызывают затраты времени и денег, не приводят к потере достоверности полученных результатов. Программное обеспечение является единственной современной технологией, качество которой зависит от тестирования.

Основной способ нахождение и исправления дефектов – блочное тестирование (unit testing), интеграционное и системное тестирование, командный анализ и анализ разработчика по написанному коду.

Во время персонального анализа написанного кода разработчик должен найти и устранить дефекты до начала тестирования. Персональный анализ написанного кода наиболее эффективен, когда он структурирован и подлежит внешней оценке. Даже на уровне анализа написанного кода нахождение ошибок является более эффективным, чем при проведении различных тестов инженерами по качеству.

Эффективность блочного тестирования составляет в среднем нахождение от двух до четырех дефектов в час и покрывает около 50% всех дефектов системы. Анализ кода (code review) находит от шести до десяти дефектов в час и покрывает около 70% дефектов системы.

Во время проведения анализа кода разработчик должен: обнаружить неправильное или странное поведение системы, выяснить какая последовательность действий приводит к неправильному поведению системы, найти, где в коде программы происходит инициация неправильного поведения системы, выяснить, к каким ошибкам может привести такое поведение. Такой способ обнаружение дефектов занимает обычно много времени. С помощью обзора и проверки кода программист может проследить за программной логикой отдельных модулей, знать, в какой части тестируемого модуля он находится во время выявлении дефекта, что программа должна делать.

Основной целью анализа кода является найти как можно больше дефектов до тестирования и выпуска версии. Для решения данной задачи программист должен использовать стандарты кодирования (coding standard), следовать полноте требований к дизайну системы, оценить и по возможности улучшить процесс анализа кода, использовать список для контроля (checklist) проведенного анализа кода.

Анализ кода будет более эффективным только в том случае, если список для контроля проведенного анализа будет построен с учетом опыта разработчика. Необходимо использовать статистические данные для выбора столбцов списка, собирать и анализировать данные во время анализа кода, сопоставить список для контроля проведенного анализа кода с существующим опытом разработчика.

В дополнение к анализу кода разработчику необходимо провести анализ дизайна системы (design review). Требованиями к данному анализу:

– созданный дизайн должен легко поддаваться анализу;

– анализ дизайна системы должен быть последовательным;

– реализованная логика должна полностью отвечать требованиям, описанным в дизайне системы.

Стадии проверки дизайна системы:

– анализ на соответствие реализованных функциональных требований к требованиям, описанным в дизайне системы;

– проверка общей структуры и потока выполнения программной логики;

– проверка логических конструкций на корректность;

– проверка на отказоустойчивость и безопасность;

– проверка правильности использования вызова функций, методов и процедур;

– проверка на корректность использования переменных, параметров методов, типов и файлов.

Для того чтобы сделать процесс анализа системы более эффективным, необходимо определить размерность оценки качества.

Размерность выхода (yield) зависит от доли дефектов в продукте, которые были найдены на данном этапе, и эффективности исправления. Размерность выхода измеряет эффективность анализа дизайна и программного кода, ревизий, компиляций версии и тестирования и рассчитывается следующим образом: размерность выхода (для фазы) = 100 * (количество найденных дефектов) / (количество найденных дефектов + количество ненайденных дефектов). Размерность выхода – это процент найденных дефектов до определенной фазы к общему количеству дефектов, найденных за фазу. Размерность выхода наиболее эффективна в случае, если разработчики и инженеры по качеству документируют все дефекты, то есть дефекты, найденные во время анализа дизайна и кода системы, тестирования и компиляции версии.

Для повышения продуктивности использования полученных значений контроль измерений должен быть доступен во время процесса измерений, например: количества частей, проанализированных за час, количества дефектов, найденных за час, количества дефектов, найденных за определенную часть фазы.

2.5 Этап выпуска и презентации

На этом этапе у студентов остается небольшое количество времени для того, чтобы подготовить презентацию и папку с документами. После подготовки этих документов подходит стадия окончательной сдачи проекта, когда все работы закончены и остается лишь презентовать и оценить проект.

Прием проекта производится преподавателем или группой преподавателей. Во время презентации проекта должны присутствовать все студенты, работающие над одним проектом. С докладом о своей части должен выступить каждый участник проекта, так как ориентироваться в системе в целом обязан каждый из них. По окончании презентации проекта ставится одна оценка всей группе, независимо от того, кто успешнее выступил. Оценивается объем выполненной работы, качество выполнения частей в целом, качество работы в команде, ответственность к поставленной задаче.

Выводы

В ходе выполнения данной работы:

– был произведен анализ текущей ситуации в области социальных сетей;

– рассмотрены существующие социальные сети, их достоинства и недостатки;

– исследована структура учебного проекта;

– процесс выполнения учебного проекта разбит на этапы;

– были распределены роли пользователей в системе и их значение;

– произведено описание архитектуры учебной социальной сети и определены необходимые требования для использования данной сети, к методу учебного проекта;

– построена модель учебного проекта в рамках концепции учебных социальной сети.

Список литературы: 1. Вингер, К. М. Разработка требований к программному обеспечению [Текст] / К. М. Вингер. – М. : Русская редакция, 2004. – 554 с. 2. Краля, Н. А. Метод учебных проектов как средство активизации учебной деятельности учащихся [Текст] : учеб. пособие / под ред. Ю. П. Дубенского. – Омск : Издательство ОмГу, 2005. – 59 с. 3. Кармайкл, Э. Быстрая и качественная разработка программного обеспечения [Текст] / Э. Кармайкл. – М. : Компьютерная литература, 2003. – 404 с. 4. Халл, Э. Разработка и управление требованиями [Текст] / Э. Халл, К. Джексон, Д. Дик. – М. : Телелоджик, 2005. – 240 с. 5. Лешек, А. М. Анализ и проектирование информационных систем с помощью UML 2.0 [Текст] / А. М. Лешек. – К. : Вильямс, 2008. – 816 с.

Поступила к редколлегии 28.10.2009

УДК 519.686.4

Використання метода навчальних проектів на базі технології соціальних мереж / О. М. Купин, Н.С. Лєсна, І.М. Келеберда // Біоніка інтелекту: наук.-техн. журнал. – 2009. – № . – С. 79-84.

Темою данної статті є дослідження і побудова моделі навчального проекту на базі технології соціальних мереж у навчальних та наукових закладів України задля інтеграції у європейський науково-технічний простір.

Метою роботи є підвищеня якості створення програмного продукту під час виконання навчального проекту.

Бібліогр.: 5 найм.

UDK 519.686.4

Using of the education project method based on the social network technology. / O. N. Kupin, N. S. Lesnaya, I. N. Keleberda // Bionics of Intelligence: Sci. Mag. – 2009. – № 2 (71). – P. 79-84.

One of the most important problem of the modern education is practice implementation of theoretical knowledge. This problem solved with education project method improve with social network conception.

Ref.: 5 items.