Большие данные и приложения

УДК 004.4.242 DOI 10.25559/SITITO.2017.3.561

Гагарин А.П., Иванов Е.В.

Московский авиационный институт (национальный исследовательский университет), г. Москва, Россия

UML-СПЕЦИФИКАЦИЯ КОМПЬЮТЕРНОЙ СРЕДЫ ДЛЯ ПРЕПОДАВАНИЯ ПРОГРАММНОЙ ИНЖЕНЕРИИ

Аннотация

Обобщённая модель передачи знаний по основным областям программной инженерии от преподавателя к учащемуся представлена в нотации языка UML как основа для проектирования и конструирования компьютерной среды, обеспечивающей как подготовку дидактических материалов, так и их применение в процессах контактного обучения программной инженерии. Рассматриваемая модель учитывает основные проблемы, встающие в процессе преподавания программной инженерии, в частности, преобладание примеров над правилами в контексте лекций, настоятельная потребность в доведении примеров программ до выполнения в ходе их демонстрации, необходимость обсуждения больших фрагментов программ, превышающих размеры экрана. Решения представлены в виде UML-спецификацию типового продукта для поддержки преподавания программной инженерии. Спецификация включает общую диаграмму прецедентов, диаграмму классов компьютеризованного учебного пособия, диаграмму классов среды для формирования таких пособий, диаграммы последовательностей, представляющих взаимодействие классов при инициализации продукта.

Ключевые слова

Программная инженерия; проектирование программ; конструирование программ; язык UML; доменная спецификация; класс; диаграмма; атрибут класса; операция класса; ассоциация; преподавание.

Gagarin A.P., Ivanov E.V.

Moscow Aviation Institute (National Research University), Moscow, Russia

UML-SPECIFICATION OF COMPUTERIZED ENVIRONMENT FOR TEACHING OF SOFTWARE ENGINEERING

Abstract

A generalized model of knowledge transferring on main areas of software engineering from a lecturer to the discipuli is represented in UML-notation as basis for design and construction of computerized environment for contact teaching in software engineering and preparation of corresponding didactical materials.The issue addresses main problems of teaching in software engineering, such as: predominance of samples over rools in a typical lecture context, imperative need of live program demos carried through execution, consideration of program samples exceeding screen size. Solutions of the probems are presented as UML-specification of a software product intended for support of teaching processes. The specification includes a general use case diagram, class diagram of online manual, class diagram of an environment providing creation of online manuals and sequence diagrams depicting cooperation of classes during initialization of the product.

Keywords

Software engineering, software design; software construction; UML; domain specification; class; diagram; class attribute; class operation; association; teaching.

Введение

Информационные ресурсы, предназначенные для поддержки учебного процесса, такие как компьютерные курсы, планы и конспекты лекций, контрольные задания и тесты, всевозможные «раздаточные материалы и другие компоненты «учебно-методического комплекса», представляют собой разновидность информационных артефактов, которые уже накопились в огромном объёме как в глобальных компьютерных сетях, так и в локальном индивидуальном пользовании и продолжают неуклонно накапливаться. Высоким темпом накопления таких ресурсов отличаются образовательные дисциплины, отражающие стремительный прогресс современных «высоких технологий», в том числе «программной инженерии» [1].

Проблемы преподавания программной инженерии

Преподавание программной инженерии имеет ряд специфических особенностей. Продуктивная деятельность в центральных областях программной инженерии: проектировании и конструировании программ – требует владения формальными языками, включая, не последнюю очередь, их прагматикой. Оно достигается не только и не столько заучиванием синтаксических и семантических правил, сколько разбором последовательности всё более сложных примеров программ и упражнениями, состоящими в составлении аналогов разобранных примеров. Правильность понимания примеров и успешность упражнений подтверждаются их выполнением в соответствующей компьютерной среде. При этом желательно, чтобы изучаемую или составляемую в порядке упражнения программу можно было незамедлительном верифицировать выполнением в режиме отладки. Различие между теоретическими лекционными и практическими лабораторными занятиями, оправданное в дисциплинах, связанных с использованием специального оборудования, в программной инженерии в большинстве случаев не имеет реальных оснований и приводит к ненужным организационным и методическим сложностям.

Особенности программной инженерии обусловливают дополнительные требования к информационным ресурсам, применяемым при её преподавании, особенно в части разработки требований к этим ресурсам, их проектировании, конструировании и тестировании. Важнейшие из этих особенностей:

- примеры преобладают над правилами в контексте лекций;
- примеры программ доводятся до компиляции и выполнения (в частности, во время лекций, а не только практических занятий и лабораторных работ);
- объяснение семантики элементов программы требует значительной части текста программы как фона (то есть, объём «темы» в количестве операторов значительно превышает объём «ремы» [2]);
- повышенная потребность в упрощающих моделях и иллюстрациях, способных повысить скорость и точность усвоения алгоритмов и структур данных;
- повышенный разброс индивидуальной скорости усвоения изучаемого материала, и вытекающая из этого повышенная потребность в средствах синхронизации образовательного процесса при групповом обучении.

Преподавание программной инженерии, несомненно, нуждается в общезначимых видах компьютерной поддержки, полезных для преподавания любых других дисциплин. К ним, в частности, относятся:

- компьютерная поддержка процессов тестирования учащихся, проверки степени усвоения преподаваемого материала, накопления и анализа данных об успеваемости;
- автоматизация подачи учебного материала, контролируемая преподавателем;
- автоматизация подготовки преподавателем индивидуальных материалов учебно-методического комплекса;
- поддержка адаптивности учебного процесса в зависимости от подготовленности контингента учащихся и форм организации учебного процесса (обучение аудиторное, дистанционное, заочное, самостоятельное и другие);
- поддержка познавательной и творческой самостоятельности учащихся.

Создание компьютерной среды для преподавания программной инженерии и разработки соответствующих информационных ресурсов актуально не только из-за особых требований, возникающих в процессе преподавания. Тематика преподавания содержательно тесно связано с компьютерными платформами, на овладение которыми направлено обучение. Эти платформы достаточно разнообразны и стремительно развиваются. Оказывается, что необходимо почти ежегодно обновлять курсы. Существенно различаются стили программирования и подходы к организации проектирования и отладки программ.

В настоящее время известно немало сред, обеспечивающих разработку учебных

компьютерных курсов по широкому кругу образовательных дисциплин, например, [3]. Ещё в начале текущего века отмечалась потребность в создания специализированных информационных курсов по программной инженерии [4]. Однако до настоящего момента ни один из таких курсов, ни среда их разработки не получили заметной известности. В то же время созданы и получили признание программные системы для ведения аудиторного преподавания по общей тематике [5].

В указанных условиях гармоничное развитие рынка информационных ресурсов по программной инженерии и правильная ориентация сообщества пользователей ставят на повестку дня выработку формализованного представления требований к среде разработки и применения таких ресурсов. Опробованным подходам к формализации такого рода является составление доменной спецификации на языке UML [6, 7], определяющей класс прикладных программ в области (домене) технического образования.

Спецификация программной поддержки учебного процесса

Предлагаемая спецификация основывается на концептуальной модели учебного процесса, в которой сформулированные выше частные и общие требованиям программной инженерии обеспечиваются целостным программным продуктом, далее упоминаемым как Продукт. Модель описывается следующими положениями.

1. Дидактическое содержание и порядок подачи материала дисциплины определяется текстовым документом в машинно-обрабатываемой форме, далее называемым Эталоном курса.

2. Продукт показывает на экранах компьютеров Эталон курса как основное пособие по изучению дисциплины в следующих режимах:

- самостоятельное изучение, без участия преподавателя – режим offline,
- дистанционное обучение (под руководством преподавателя, но без аудиовизуального контакта с ним) – режим onlineDistant,
- обучение в аудитории под руководством преподавателя (с учётом возможности аудиовизуального контакта с ним в интересах ведения учебного процесса) – режим onlineLocal.

3. Продукт поддерживает испытания учащихся в целях закрепления полученных знаний, проверки готовности к очередному этапу обучения и оценки успеваемости.

4. Для online-режимов (режимов onlineLocal и onlineDistant) Продукт обеспечивает обратную связь, от учащегося к преподавателю, в виде вопросов и результатов выполнения испытательных работ.

5. Для режима onlineLocal в условиях группового обучения Продукт обеспечивает синхронизацию подачи материала преподавателем, получение материала учащимися и получение преподавателем ответно-вопросной информации от учащихся.

Эти принципы выражают роль Продукта в учебном процессе и определяют минимальные требования к его функциям. Для более полного удовлетворения потребностям преподавания программной инженерии для Продукта постулируются следующие дополнительные принципиальные требования.

6. Продукт должен поддерживать ведение индивидуальных конспектов курса учащимися и протоколов учебных занятий преподавателем.

7. Эталон курса должен включать в себя базовый план ведения учебных занятий, согласно которому преподаватель осуществляет через Продукт подачу учебного материала. Кроме того, обеспечивается возможность отклонений от базового плана занятий, управляемых преподавателем, в целях обсуждения возникающих вопросов, дополнительного разъяснения трудных мест, самостоятельного выполнения учащимися упражнений. При этом продукт обеспечивает возврат к базовому плану с сохранением дополнительных материалов в конспектах учащихся и в протоколе занятий преподавателя.

8. Продукт позволяет преподавателю, а также учащимся в индивидуальном порядке оперативно, во время занятия использовать сторонние, не заключённые в Эталонном курсе материалы из информационных сетей и других источников. Эти материалы могут факультативно включаться в конспекты учащихся и в протокол занятий преподавателя.

9. Продукт обеспечивает в ходе учебного занятия редактирование, компиляцию и выполнение текстов программ, входящих в Эталон курса или привлечённых с целью демонстрации или выполнения испытательной работы. Продукт открыт для подключения к нему редакторов программ и систем программирования, необходимых для демонстраций и упражнений.

10. Эталон курса может включать элементы, предполагающие анимацию в процессе их показа, а Продукт должен обеспечивать показ по шагам анимации под управлением преподавателя.

11. Продукт открыт по отношению к используемым Эталонам курсов и предоставляет преподавателю возможность создавать и

редактировать Эталоны курсов, подходящие для использования в его среде.

12. Как любая программа, коллективное использование которой предполагает соблюдение от пользователей определённой дисциплины, Продукт должен обеспечивать аутентификацию и авторизацию пользователей – учащихся и преподавателей – в соответствии с их ролями и полномочиями в учебном процессе.

Далее рассматриваются архитектурные решения, направленные на удовлетворение Продуктом сформулированных выше требований. Эти решения относятся к следующим аспектам проектирования Продукта:

- внешние, обращённые к пользователю функции и режимы работы Продукта,
- структура Эталона курса,
- обеспечение разработки Эталона Курса,
- организация продвижения по курсу,
- вход пользователя в операционную среду Продукта и её настройка для ведения учебного процесса.

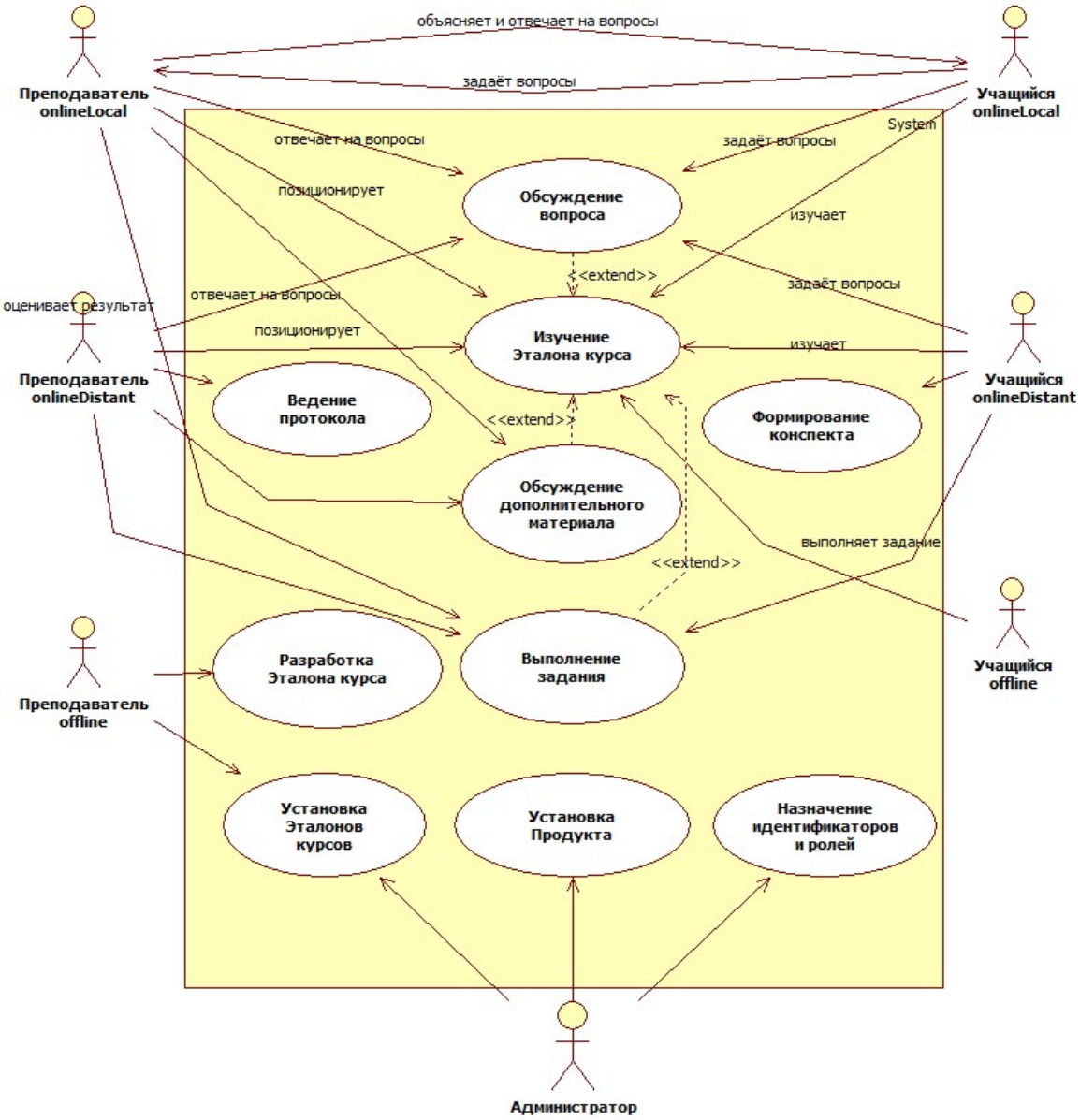

Рисунок 1. Диаграмма прецедентов (use case) Продукта

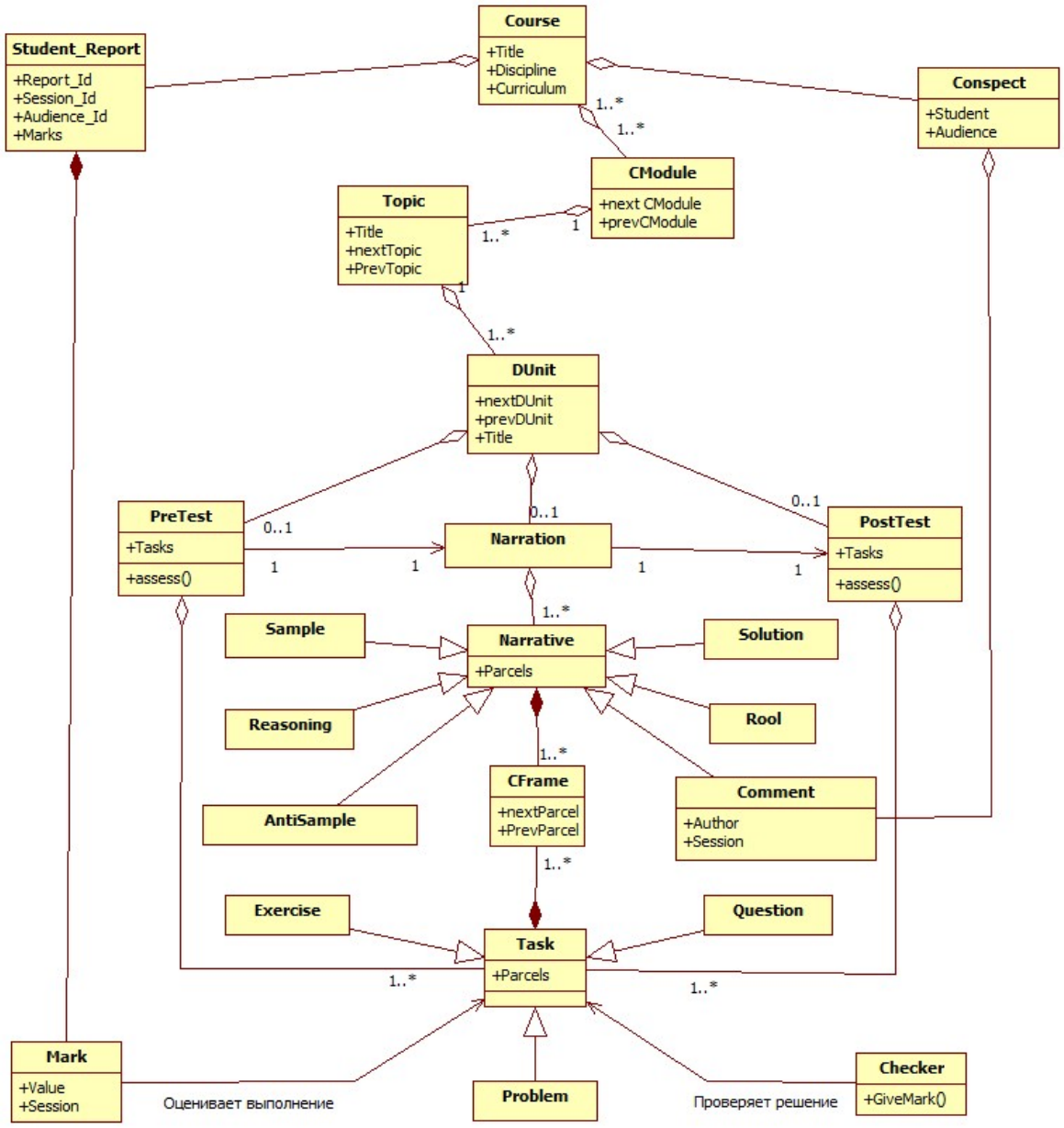

Рисунок 2. Диаграмма классов, представляющая структуру Эталона курса

Режим функционирования Продукта определяется значениями параметров Mode и Role. Непрерывный промежуток времени функционирования, в течении которого значения этих параметров не изменяются, называется «сессией». В спецификации сессия представлена классом Session. Значения параметров и смысл их сочетания показаны в Табл. 1.

Прецеденты, то есть крупные независимые функции, которые продукт должен выполнять, обслуживая различные категории своих пользователей, показаны на Диаграмме прецедентов на Рис.1.

Таблица 1. Режимы и роли функционирования Продукта

Mode\Role	Учащийся	Преподаватель
offline	Самостоятельная работа учащегося по освоению или повторению курса	Подготовка курса
onlineLocal	Учёба в условиях аудиоконтакта с преподавателем в аудитории или на телеконференции	Преподавание в условиях аудиоконтакта с учащимся в аудитории или на телеконференции

onlineDistant	Учёба при условии взаимодействия с преподавателем только через программу	Преподавание при условии взаимодействия с учащимся только через программу

Параметры onlineLocal и onlineDistant требуют, чтобы терминалами преподавателя и учащегося находились в одной терминальной сети или одной компьютерной сети. Связь между терминалами студентов не предусматривается и не обеспечивается.

Структуру Эталона курса в целом можно представить диаграммой классов, представленной на Рис.2.

Эталон курса образован последовательностью модулей (в спецификации – класс CModule), обладающий такой степенью смысловой самостоятельности, что его можно переносить из одного курса в другой. Модуль которых распадается на последовательность тем (класс Topic), а тема – в свою очередь – на последовательность «дидактических единиц» (в спецификации – класс DUnit). Дидактическая единица является минимальной единицей курса для усвоения учащимся и проверки степени этого усвоения.

Дидактическая единица предваряется проверкой готовности учащегося к усвоению её содержания. Проверка осуществляется выполнением последовательности заданий (класс PreTest). Содержание представлено в модели классом Narration – последовательностью нарративных единиц (класс Narrative), подлежащих усвоению и не содержащих заданий, требующих решения задач и ответа. Дидактическая единица завершается проверкой усвоения её содержания путём выполнения последовательности заданий (класс PostTest). Задание в составе PreTest и PostTest представлена классом Task.

Частными случаями нарративной единицы являются: «правило», «пример», «анти-пример» – то есть пример, показывающий неправильный способ действия, «обоснование», «решение», «замечание». В модели они представлены классами Rool, Sample, AntiSample, Reasoning, Solution, Comment. Частными случаями задания являются «вопрос», «упражнение», «проблема» (классы Question, Exercise, Problem, соответственно).

Для автоматизированной обработки результатов усвоения, в классах PreTest и PostTest предусматривается процедура Assess. Для каждой задачи она вызывает метод GiveMark класса Checker. Этот метод автоматически или путём обращения к преподавателю вырабатывает «оценку» – объект класса Mark. Их последовательность образует «табель» – объект класса Student_Report, который сохраняется для каждого учащегося, в каждой сессии каждой аудитории. В аудитории также храниться последовательность отчётов сессий – объектов класса Session_Report. Этот объект хранит идентификатор и дату сессии, а также идентификатор последнего изучавшегося модуля и пройденного в нём шага.

Работу за компьютером можно представить, если отвлечься от содержания обрабатываемой информации, как обмен «сообщениями»: работник за компьютером вводит сообщения посредством клавиатуры и манипулятора, а компьютер выводит сообщение, изменяя содержимое окна на экране. Границы между сообщениями определяются промежутками относительной стабильности содержания окна («паузами»). Анимации в учебном курсе ограничены. Чтобы не смешивать сообщения, относящиеся к содержанию курса, с сообщениями операционной среды, сообщения курса называются «кадрами» (в спецификации – класс CFrame).

Преподаватель, пользуясь Продуктом, выдаёт поток кадров, часть которых требует от учащегося только сигнала, что содержание полученной посылки усвоено и он готов к работе над следующим кадром, а часть представляет собой задания, в ответ на каждую из которых учащийся обязан направить ответную посылку с решением. Кроме того, учащийся может выдавать посылки-вопросы преподавателю, получать дополнительные сообщения устно в условиях аудитории или видеоконференции, а также формировать собственный «конспект» курса (класс Conspect), записывая свои комментарии, сообщения от преподавателя, поступающие вне компьютерной системы, или сведения из справочников. Учащийся вводит эти дополнения (класс Comment), и они привязываются к позициям внутри курса, при изучении которых они возникли.

Сессия в режиме offline для роли Teacher предназначена для подготовки курсов. Главное окно в этом случае (класс Workshop) создание нового курса, выбор курса из существующих для редактирования, добавление к нему или удаление из него элементов: модулей, шагов и посылок. Кроме того, это окно позволяет пользоваться для формирования контента всеми ресурсами платформы.

Модель действий преподавателя при разработке компьютерного курса показана на диаграмме классов на Рис.3.

Преподавание в аудитории (режим onlineLocal)

предлагается рассматривать как последовательность эпизодов, в течение каждого из которых преподаватель объясняет или задаёт вопросы студентам на основе материала, представленного на экранах компьютеров. Границы эпизодов формально определяются заменой или трансформацией этого материала.

Материал представляется, как минимум, в трёх окнах: главное окно (Main Window, или MW), окно просмотра (Browsing Window, или BW) и служебном окне (Service Window, или SW). Главное окно предназначено для демонстрации заранее заготовленного материала курса. В этом окне находится кадр курса, являющаяся предметом текущего рассмотрения на лекции. Окно просмотра предназначено для обеспечения ретроспекции – вспомогательного просмотра уже «пройденных» кадров курса. При переходе в главном окне на следующий кадр предыдущий кадр главного окна автоматически отображается в окне просмотра, если иное не определено преподавателем. Сервисное окно предназначается для работы с информацией, не входящей в заранее заготовленный материал курса: справки из Интернета, сообщения, работа с системами программирования и другое. Все три окна в модели представлены классом CView.

Процессом смены эпизодов занятия руководит преподаватель. До перехода к новому эпизоду преподаватель формирует материал следующего эпизода в окнах своего компьютера. Когда материал сформирован, преподаватель инициирует функцию Synchronize, в результате выполнения которой у учащихся и у преподавателя в окнах компьютеров устанавливается идентичное содержание.

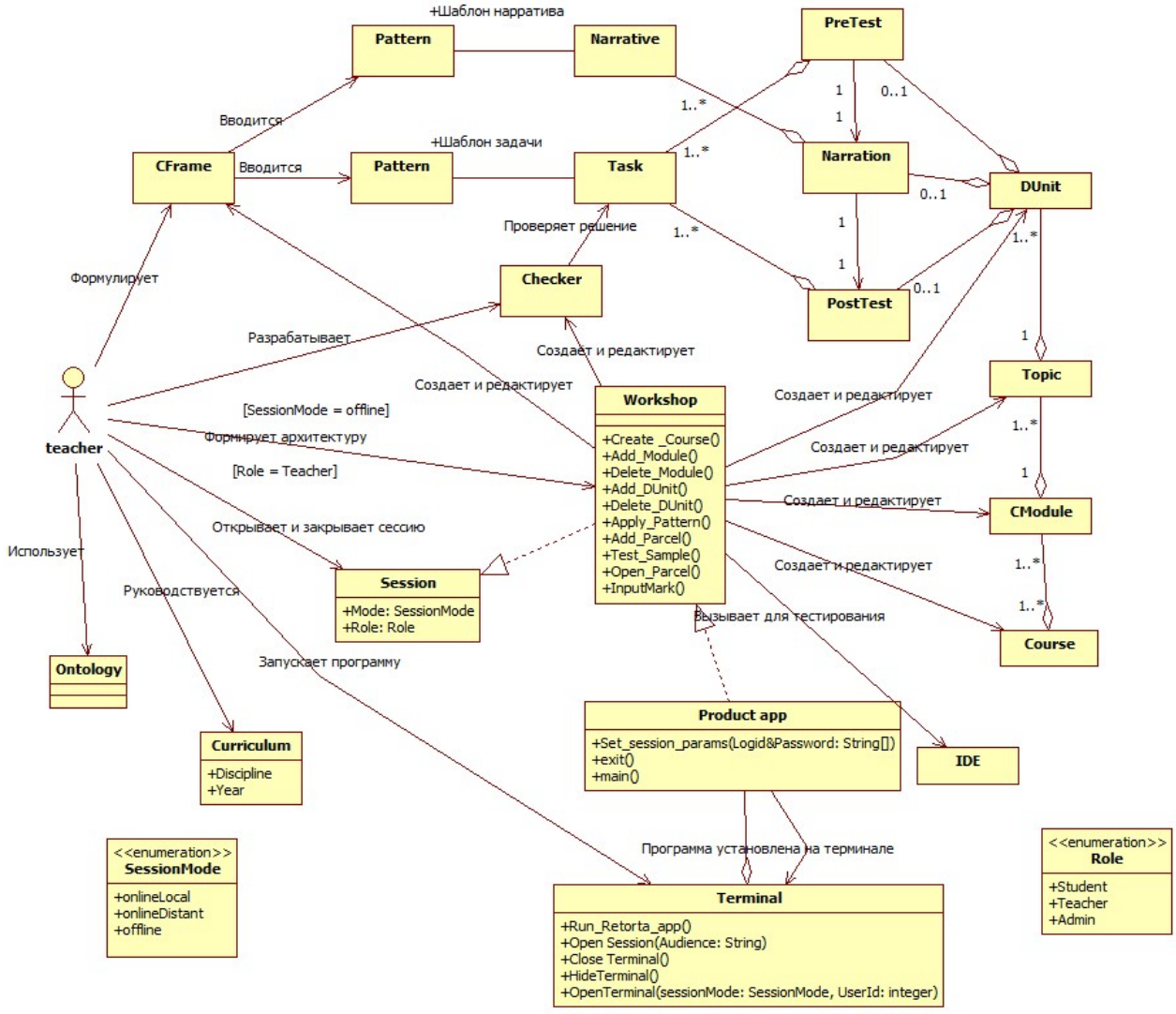

Рисунок 3. Диаграмма классов, представляющая разработку Эталона курса

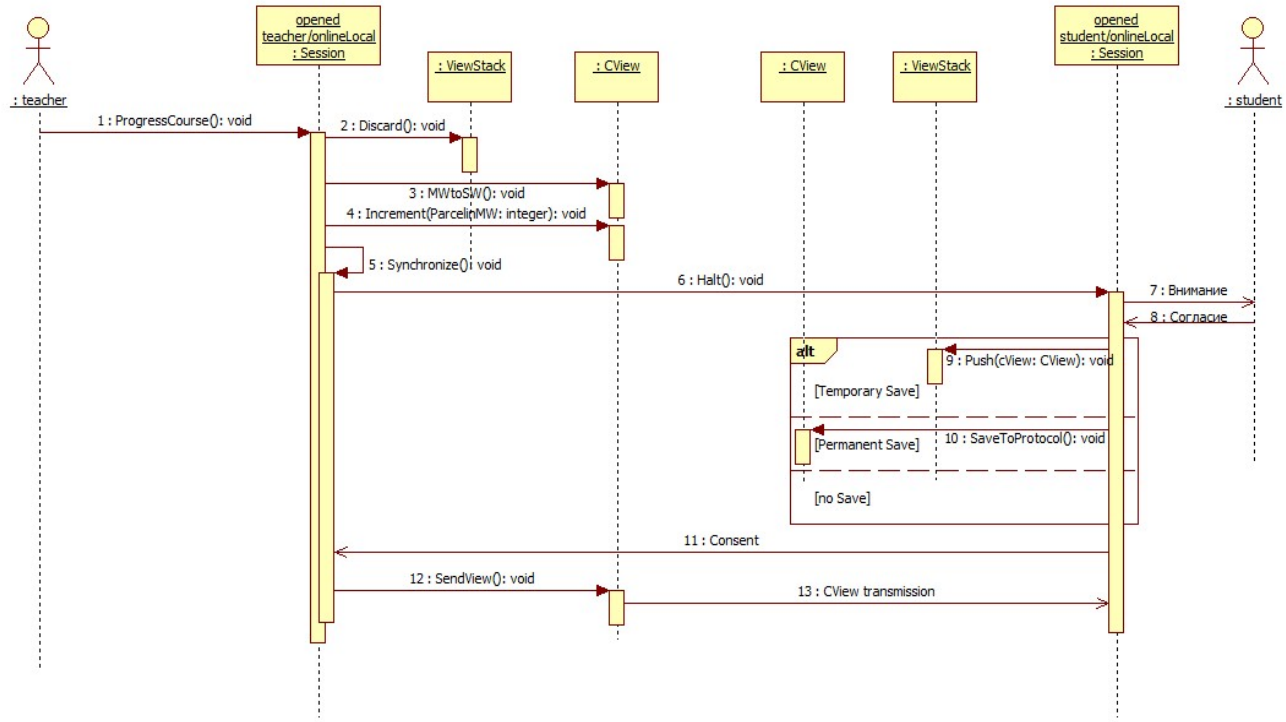

Рисунок 4. Диаграмма последовательностей, представляющая продвижение по курсу

В течение эпизода преподаватель может изменять содержание любого из окон, а учащийся – вспомогательного окна и окна просмотра. Содержание всех трёх окон может при этом быть сохранено в специальном стеке – классе ViewStack – командами Push и Pope. Предусмотрена команда восстановления первой сохранённой копии окон и аннулирования содержимого стека. Это механизм сохранения предусмотрен для обеспечения ограниченной автономии действий преподавателя и учащихся и возможности преподавателю оперативно отклоняться от запланированной в курсе последовательности рассмотрения кадров.

В частности, учащийся имеет возможность задержать синхронное обновление материалов в окнах. Для этого при поступлении сигнала синхронизации от преподавательского компьютера учащийся получает от своего компьютера и может сохранить текущее содержимое сервисного окна и окна просмотра в стеке прежде, чем согласиться на синхронизацию. Синхронизация задерживается до поступления этого согласия.

Учащийся в любой момент может послать сообщение преподавателю с вопросом или с пожеланием остановить ход лекции. Чтобы организовать ответ, преподаватель может воспользоваться окнами своего компьютера и их отсылкой учащемуся в качестве ответа, сохранив в стеке состояние окон момента поступления запроса.

Все компьютеры – учащихся и преподавателя – показывают в своей основе один и тот же материал. Этой основой служит курс, подготовленный и преподавателем.

На Рис. 4 показана диаграмм классов, иллюстрирующая рассмотренные действия преподавателя при продвижении по курсу.

Действия пользователя – преподавателя или учащегося по редактированию окон в пределах эпизода показаны на Рис. 5.

Пользователю – как преподавателю, так и учащемуся – предоставляются возможности работы в любом из трёх окон, поддерживаемых Продуктом во время сессии: MW, BW и SW. Их назначение описано выше, в связи диаграммы на Рис. 4. Следует отметить, что эти окна не обязательно реализовать окнами операционной системы. Они могут представлять собой отдельные зоны главного окна Продукта.

Операции Focus(), Select(), Navigate(), Animate() на диаграмме Рис.5 показаны для окна BW, но они определены для всех трёх окон.

Операция Focus() выражает выбор окна для выполнения с ним остальных операций..

Операция Select() представляет группу операций ассоциативного поиска и выбора элемента некоторой коллекции для отображения в окне. В окнах MW и BW основным объектом выбора является кадр в Эталоне курса или в индивидуальном конспекте учащегося.

Операция Navigate() представляет группу операций выбора элемента коллекции по индексу в коллекции или по номеру позиции относительно текущего положения. Предметом выбора могут выступать кадры курса, строки в листинге программы, строки таблиц. Частным случаем этой операции являются обычные операции Next() и Previous().

Операции группы Animate() предназначены для управления демонстрацией анимированных кадров, например, слайдов в стиле PowerPoint. Пользователь операциями этой группы вызывает пошаговую трансформацию содержания кадра, демонстрируя, например, шаги выполнения программы или шаги её проектирования.

Диаграмма на Рис.6 показывает взаимодействие классов при входе в Продукт и подготовке к открытию сессии. Вход осуществляется одинаково для преподавателя, учащегося и администратора. Класс Sentinel отвечает за проверку идентификаторов пользователей и их паролей. При регистрации идентификатор связывается с ролью пользователя. Затем пользователь указывает роль, в которой он намеревается действовать в среде Продукта. Если он имеет права на эту роль, то открывается виртуальный терминал, представляющий пользователя в определённой роли. Такой терминал представлен классом Terminal. Конкретный пользователь может открыть несколько терминалов для разных ролей и переходить между ними без повторной авторизации.

В среде открытого терминала пользователь может открывать сессии, вводя идентификатор «Аудитории» (класс Audience).

Аудитория формируется администратором и охватывает конкретную группу пользователей, как правило, преподавателя и коллектив учащихся, а также изучаемую дисциплину. Изучаемая дисциплина представляется идентификатором Эталона курса.

Процесс открытия сессии показан на Рис.7.

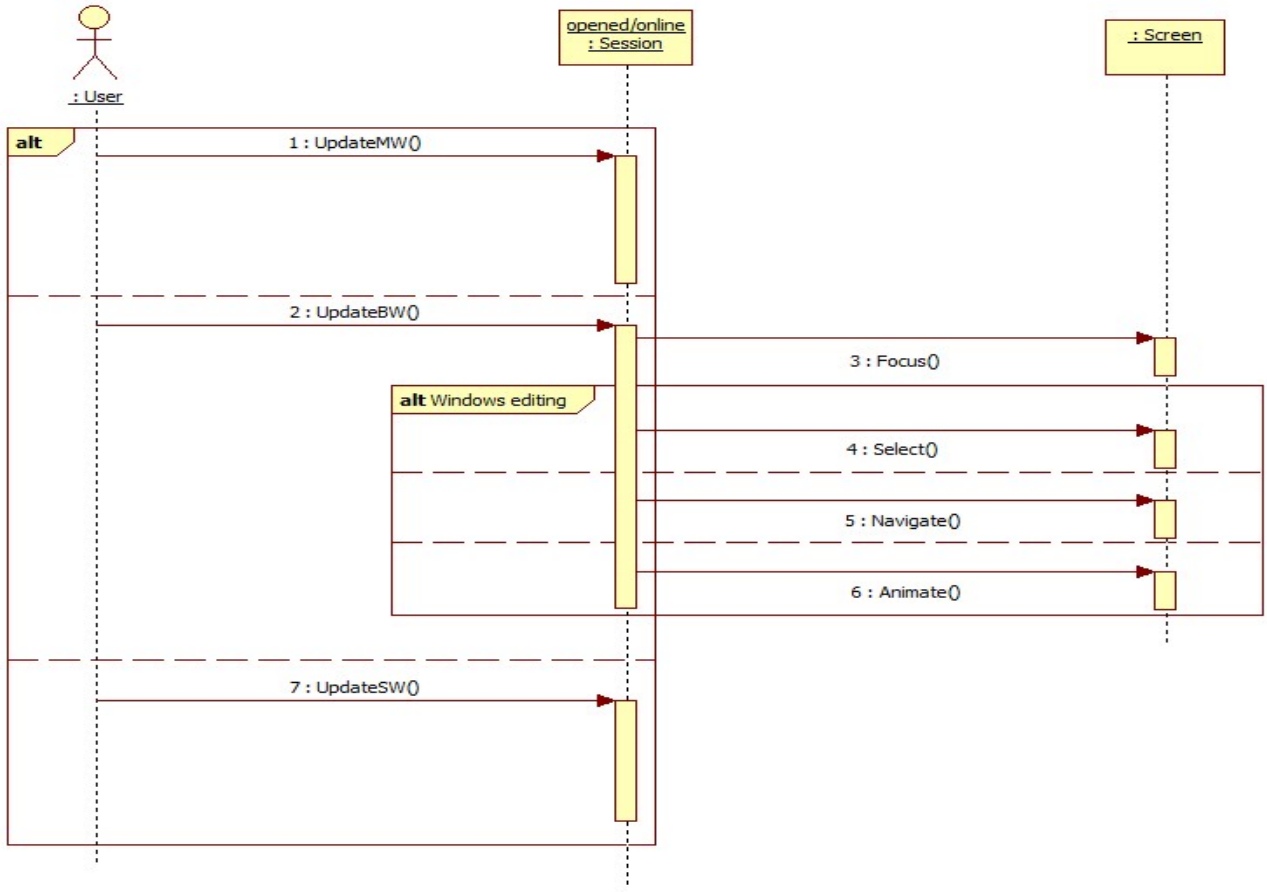

Рисунок 5. Диаграмма последовательностей, представляющая трансформацию в окнах

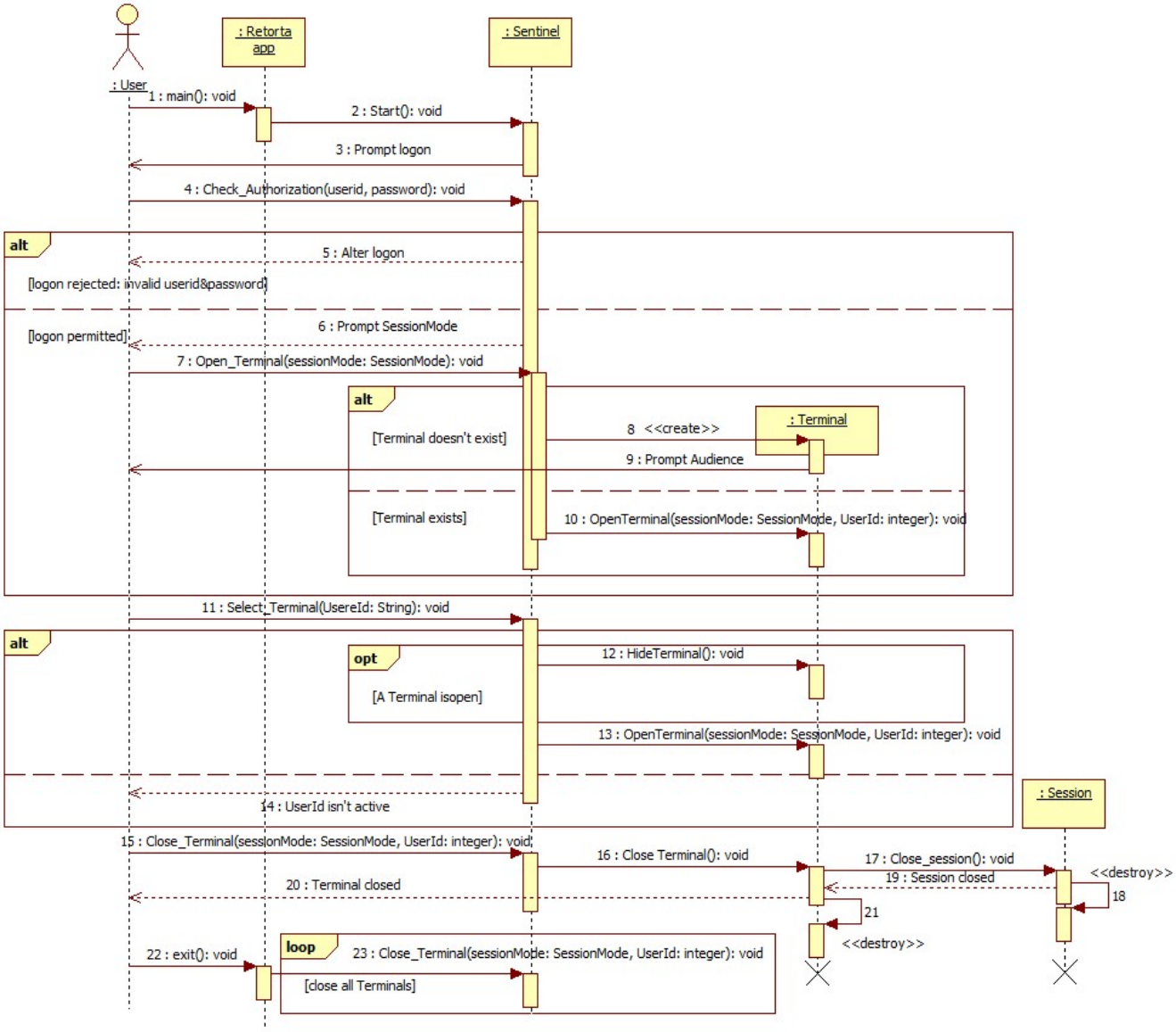

Рисунок 6. Диаграмма последовательностей, представляющая запуск Продукта

Заключение

Предложенные спецификации не накладывают жестких ограничений на способы их реализации в конкретных программных продуктах. В то же время следование этим спецификациям или, хотя бы их учёт создают основу для обеспечения их совместимости в отношении дидактического материала, унификации организационных форм ведения учебного процесса и практических навыков пользователей – преподавателей и учащихся.

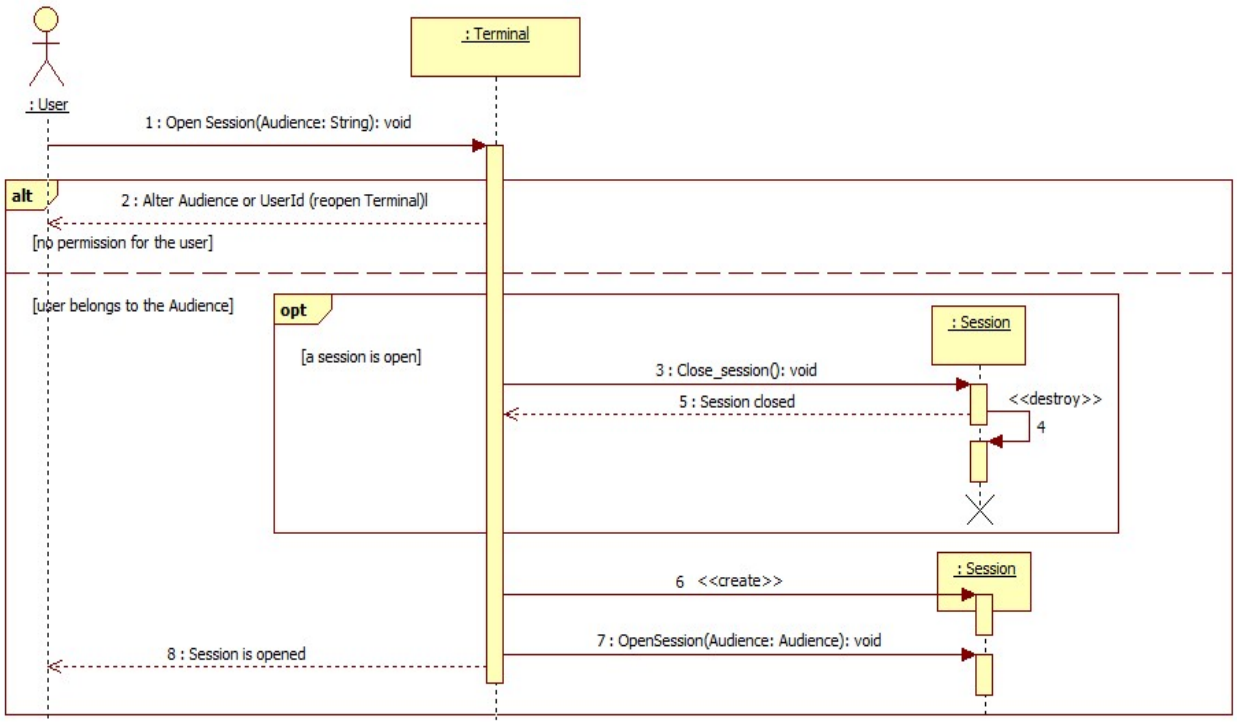

Рисунок 7. Диаграмма последовательностей, представляющая открытие сессии

Литература

1. Guide to the Software Engineering Body of Knowledge Version 3.0 SWEBOK Editors: Pierre Bourque, Richard E. (Dick) Fairley, 2014
2. Шевякова В. Е. Актуальное членение предложения // Лингвистический энциклопедический словарь. – М.: СЭ, 1990. — С. 22—23. [Электронный Ресурс] URL: http://tapemark.narod.ru/les/022f.html
3. Moodle –open-source learning platform [Электронный Ресурс] URL: https://moodle.org/
4. Ernst-Erich Doberkat[a], Gregor Engels[b] Software Engineering and eLearning: The MuSofT Project Dept. of Computer Science, University of Dortmund, Germany URL: https://eleed.campussource.de/archive/2/201
5. NETOP Vision [Электронный Ресурс] URL: https://www.netop.com/edu.htm
6. Introduction to OMG Specifications [Электронный Ресурс] URL: http://www.omg.org/gettingstarted/specintro.htm#DomainFacilities
7. OMG Unified Modelling Language (OMG UML). Version 2.5. [Электронный Ресурс] URL: http://www.omg.org/spec/UML/2.5/PDF

References

1. Guide to the Software Engineering Body of KnowledgeVersion 3.0 SWEBOK Editors: Pierre Bourque, Richard E. (Dick) Fairley, 2014
2. Shevjakova V. E. Aktualnoe chlenenie predlozhenija // Lingvisticheskij entsiklopedicheskij slovar'. — M.: SE 1990. — P. 22—23. [Электронный Ресурс] URL: http://tapemark.narod.ru/les/022f.html
3. Moodle –open-source learning platfor [Электронный Ресурс] URL: https://moodle.org/
4. Ernst-Erich Doberkat[a], Gregor Engels[b] Software Engineering and eLearning: The MuSofT Project Dept. of Computer Science, University of Dortmund, Germany [Электронный Ресурс] URL: https://eleed.campussource.de/archive/2/201
5. NETOP Vision URL: https://www.netop.com/edu.htm
6. Introduction to OMG Specifications [Электронный Ресурс] URL: http://www.omg.org/gettingstarted/specintro.htm#DomainFacilities
7. OMG Unified Modelling Language (OMG UML). Version 2.5. [Электронный Ресурс] URL: http://www.omg.org/spec/UML/2.5/PDF

Поступила: 26.10.2017

Об авторах:

Гагарин Андрей Петрович, кандидат технических наук, профессор по кафедре информатики, Московский авиационный институт (национальный исследовательский университет), gagarin_ay@outlook.com

Иванов Елисей Викторович, магистрант, Московский авиационный институт (национальный исследовательский университет), h4ck3r1121@yandex.ru

Note on the authors:

Gagarin Andrey P., Candidate of Engineering Sciences, Professor on Informatics, Moscow Aviation Institute (National Research University), gagarin_ay@outlook.com

Ivanov Elissey V., magistrant, Moscow Aviation Institute (National Research University), h4ck3r1121@yandex.ru