УДК 519.24

С.Ц.МАНЖИКОВА, С.В.БОРОДКИН, И.И.НАЗАРОВ
E-mail:ksucta@elcat.kg.

ЭЛЕКТРОННЫЙ ЛАБОРАТОРНЫЙ ПРАКТИКУМ ПО ДИСЦИПЛИНЕ «CASE-ТЕХНОЛОГИИ РАЗРАБОТКИ ПО» С ИСПОЛЬЗОВАНИЕМ ПАТТЕРНОВ ПОВЕДЕНИЯ

Электрондук-лабораториялык иштер үчүн паттерндердин иллюстрациялык бир катар өзгөчөлүктөр менен UML 2.0 моделине унифициалдуу (стандарттуу) тил үчүн (бир) акыркы версиясы колдонулган атайын курс иштелип чыккан. Лабораторлук иштер мультимедиялык коштоо менен камсыздалган.

Разработан специальный курс электронных лабораторных работ, иллюстрирующий возможности и особенности ряда паттернов, для чего применена последняя версия (стандарт) унифицированного языка моделирования UML 2.0. Лабораторные работы обеспечены мультимедийной поддержкой.

We developed a special course of electronic laboratory works, illustrating the possibilities and features of the patterns. To do this the latest version (standard) of the Unified Modeling Language UML 2.0 was applied. Laboratory works are provided by multimedia support.

Компетентностный подход /1, 2/ как основная составляющая инновационных образовательных технологий современного вуза неразрывно связан с применением электронных средств обучения будущих специалистов высокой квалификации в любой области знания. Особую роль электронное обучение играет в процессе профессиональной подготовки инженеров-программистов. Это обусловлено высокими темпами развития технологий разработки программных систем различного функционального назначения и мощности, и, следовательно, объективным является требование отобразить результаты этого развития в учебном процессе с применением адекватных программных средств.

КРСУ осуществляет эффективные методы обучения современным технологиям программирования. В последние годы существенное внимание уделяется CASE-технологиям разработки ПО (CASE – Computer Aided Software/System Engineering). Успешное применение этих технологий на практике позволяет в несколько раз увеличить производительность труда программиста, одновременно гарантируя создание корректного, работоспособного и надежного ПО, обладающего свойствами модифицируемости и адаптируемости.

Очередной задачей, которая решается на этом пути, является представление в учебном процессе CASE-программного продукта нового поколения – Visual Studio 2010 – и использование его в сочетании с технологией применения паттернов проектирования /3/ программного обеспечения и Unit-тестирования. С этой целью был разработан специальный курс электронных лабораторных работ, иллюстрирующий возможности и особенности ряда паттернов, для чего применена последняя версия (стандарт) унифицированного языка моделирования UML 2.0. Лабораторные работы обеспечены мультимедийной поддержкой.

Концептуальная модель программной системы представлена на рис.1. С системой взаимодействует **Пользователь.** Начальными в работе с системой являются варианты ее использования **Вход в электронный лабораторный практикум** и **Поиск по электронному лабораторному практикуму.** Эти варианты использования обладают иерархической структурой, которая на этом верхнем уровне отображается в виде **Списка лабораторных работ.**

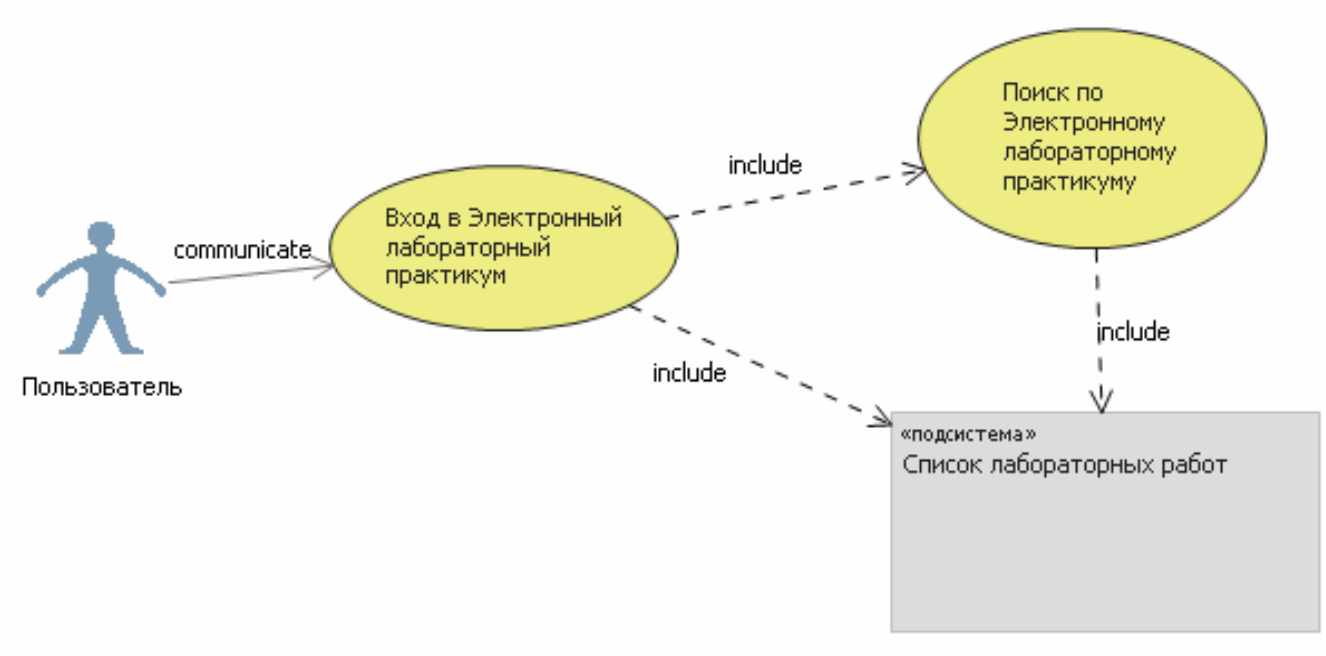

Рис. 1. Диаграмма вариантов использования электронного лабораторного практикума

Декомпозиция подсистемы Список лабораторных работ представлена на рис.2.

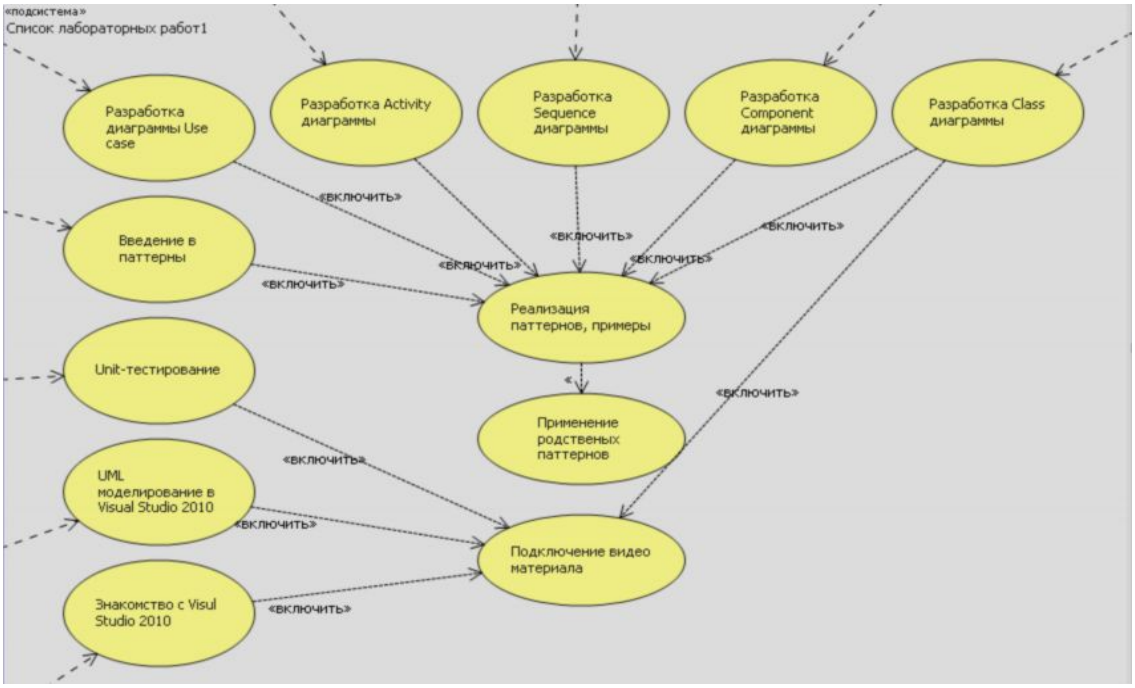

Рис. 2. Диаграмма вариантов использования подсистемы **Список лабораторных работ**

Здесь отображены следующие режимы пользования электронным лабораторным практикумом:

– **Знакомство с Visual Studio 2010** (лабораторная работа, посвященная Visual Studio 2010),

– **UML моделирование в Visual Studio 2010** (лабораторная, посвященная UML-моделированию в Visual Studio 2010),

– **Unit-тестирование** (лабораторная, посвященная Unit-тестированию),

– **Введение в паттерны** (лабораторная, содержащая вводную информацию о паттернах),

– **Разработка диаграммы Use Case** (лабораторная, посвященная построению диаграммы Use Case в Visual Studio 2010),

– **Разработка Activity диаграммы** (лабораторная, посвященная построению диаграммы Activity в Visual Studio 2010),

– **Разработка Sequence диаграммы** (лабораторная, посвященная построению Sequence диаграммы в Visual Studio 2010),

– **Разработка Component диаграммы** (лабораторная, посвященная построению Component диаграммы в Visual Studio 2010),

– **Разработка Class диаграммы** (лабораторная, посвященная построению Class диаграммы в Visual Studio 2010),

– **Примеры реализации паттернов** («исходники» кода, используемые в лабораторном практикуме),

– **Родственные паттерны** (описание паттернов, являющихся родственными паттернам, представленным в курсе),

– **Видео-материал** (материал, содержащий информацию в видео-формате).

Каждый из приведенных на рис.2 вариантов использования (элементов Use Case) фактически представляет отдельную лабораторную работу, которая должна быть выполнена в соответствии со своим сценарием (спецификацией элемента Use Case).

Порядок пользования электронным лабораторным практикумом показан с помощью диаграммы деятельности (рис. 3).

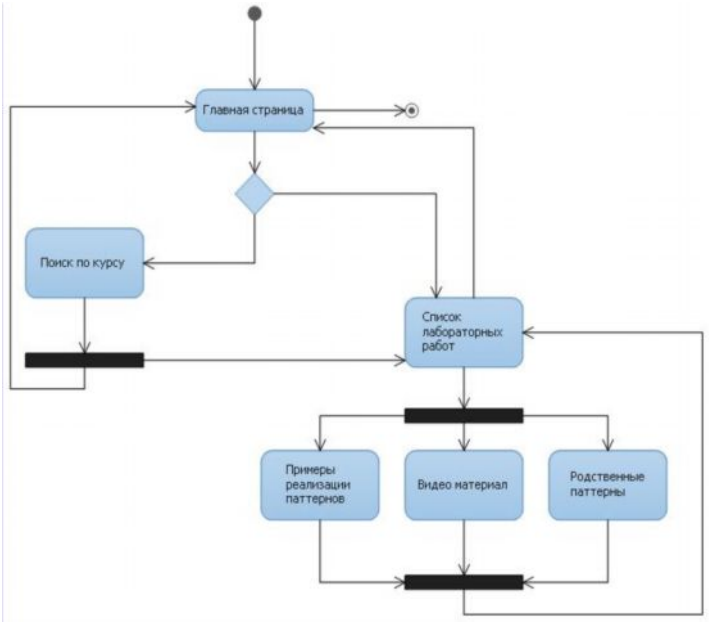

Рис. 3. Диаграмма деятельности пользования электронными лабораторными работами

Для создания данного электронного лабораторного практикума использовались следующие средства и среды реализации:

- HTML
- CSS
- Java
- Flash
- Java Script
- PHP
- Visual Studio 2010
- Camtasio Studio

В качестве языка гипертекстовой разметки был выбран HTML. Для создания и обработки стилей был использован CSS, так как это позволяет упростить процесс описания и создания тегов в HTML.

При создании процедур поиска использовался PHP, так как главным положительным фактором языка PHP является его практичность. PHP предоставляет программисту средства для

быстрого и эффективного решения поставленных задач. Практический характер PHP обусловлен пятью важными характеристиками: традиционностью, простотой, эффективностью, безопасностью, гибкостью. Существует еще одна «характеристика», которая делает PHP особенно привлекательным: он распространяется бесплатно, причем с открытыми исходными кодами.

Для реализации паттернов применялись объектно-ориентированные языки Java и JavaScript. Для визуализации паттерна Наблюдатель использовалась мультимедийная платформа Flesh.

Для построения UML-диаграмм и для создания UNIT-тестов использовалась среда Visual Studio 2010, так как здесь поддерживается стандарт языка UML 2..

При создании видеокурса использовалась утилита для записи изображения Camtasia Studio v7.0.0 в связи с тем, что данный продукт имеет широкие возможности и занимает одно из ведущих мест по работе с мультимедиа.

Разработка электронных обучающих систем требует предварительного структурирования информации, которая предлагается студенту для усвоения и приобретения соответствующих практических навыков. Каждая из лабораторных работ имеет следующую структуру:

– Вводная информация.

– Паттерн как «кирпичик ПО»:

- Диаграмма Use Case (вариантов использования),
- Sequence диаграмма (Диаграмма последовательности),
- Component диаграмма (Диаграмма компонентов),
- Class диаграмма (Диаграмма классов),
- Activity диаграмма (Диаграмма деятельности).

– Пример реализации паттерна:

- Демонстрация элементов кода.

– Достоинства и недостатки.

– Родственные паттерны.

– Задания.

Все эти пункты студенту рекомендуется выполнять последовательно.

Ниже приводятся примеры представления паттерна **«Абстрактная Фабрика»** в электронном лабораторном практикуме и его применение в процессе разработки ПО. **«Абстрактная Фабрика»** – паттерн, порождающий объекты. Предоставляет интерфейс для создания группы объектов, бизнес-объектов участников всей системы, использующихся далее в рамках всего приложения. Смысл состоит в том, что их конкретные классы не объявляются, позволяя, таким образом, легко заменять их в дальнейшем как отдельные объекты, так и все взаимосвязанное семейство. В этом паттерне используются следующие классы:

AbstractFactory(WidgetFactory) – абстрактная фабрика. Объявляет общий интерфейс для операций создания абстрактных объектов – продуктов. Далее в приложении вместо абстрактных будут создаваться уже конкретные объекты какого-либо одного семейства.

ConcreteFactory*[N]*(Motif WidgetFactory, PMWidgetFactory) – конкретная фабрика (класс). Определяет, реализует операции, создающие все объекты одного конкретного семейства.

AbstractProduct*[A-Z]*(Window, ScrollBar) – абстрактный продукт (класс). Определяет общий интерфейс для типа объекта-продукта. A-Z в данном случае – множество таких продуктов, составляющих абстрактное семейство.

ConcreteProduct[A-Z][N]Motif Window, Motif ScrollBar) – конкретный продукт (класс). Определяет конкретный объект-продукт типа [A-Z], создаваемый соответствующей конкретной фабрикой [N].

Client–Клиент. Другой программный модуль, фрагмент кода, дающий команды на получение конкретного семейства продуктов, пользуясь только известными интерфейсами классов AbstractFactory и AbstractProduct. **Клиент(**Client**)** создает единственный экземпляр ConcreteFactory, с помощью которого получает, когда ему нужно, все объекты соответствующего семейства. Поэтому для создания других объектов, выполняющих те же функции каким-нибудь несколько иным способом, нужно лишь инстанцировать, создать другую ConcreteFactory.

Разработка Component диаграммы, соответствующей паттерну Абстрактная Фабрика

Диаграмма **Component**, в отличие от ранее рассмотренных диаграмм, описывает особенности физического представления системы. Диаграмма компонентов позволяет определить архитектуру разрабатываемой системы, установив зависимости между программными компонентами, в роли которых может выступать исходный, бинарный и исполняемый код. Во многих средах разработки модуль или компонент соответствует файлу. Пунктирные стрелки, соединяющие модули, показывают отношения взаимозависимости, аналогичные тем, которые имеют место при компиляции исходных текстов программ. Основными графическими элементами диаграммы компонентов являются **компоненты**, **интерфейсы** и **зависимости между ними**. На данной диаграмме представляются (программные) компоненты паттерна **«Абстрактная фабрика».** Благодаря диаграмме компонентов можно визуально представить все классы, входящие в паттерн **«Абстрактная фабрика»** и взаимодействующие в процессе его функционирования.

В ходе выполнения лабораторных работ студентам предлагается разработать ту или иную программу без применения паттернов и с их использованием, а затем оценить целесообразность и достоинства последнего варианта. В результате такого подхода приобретается критическое отношение к различным профессиональным приемам проектирования ПО.

На рис. 4 представлена главная страница электронного практикума, где можно визуально ознакомиться с интерфейсом всего курса, увидеть весь список лабораторных работ и вводную информацию к курсу.

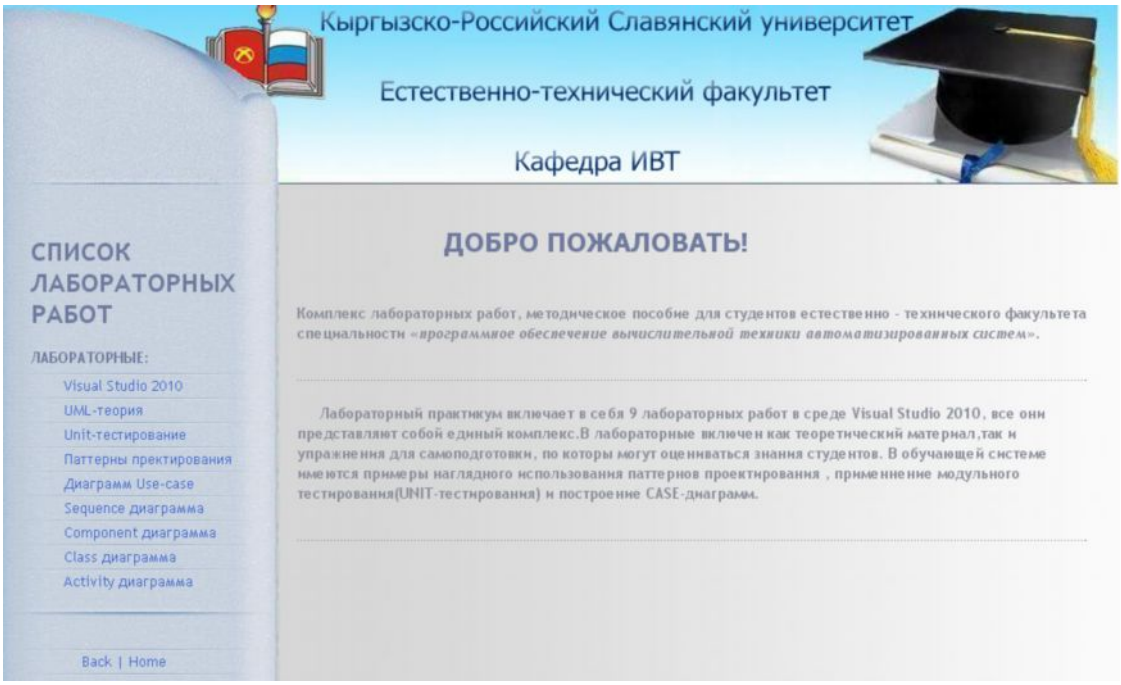

Рис. 4. Главная страница электронного лабораторного практикума

В литературе /3/ отмечается, что обучение проектированию программных систем само по себе является одним из самых трудных видов преподавательской деятельности и потому не может быть абсолютно успешным применительно ко всем студентам. Однако разработанный электронный лабораторный курс может значительно усилить такую характеристику компетентностного подхода, как индивидуализация обучения. Он позволяет преподавателю определить наиболее подходящую траекторию обучения, а студенту – траекторию самостоятельного освоения материала.

Предлагаемая разработка учитывает современные тенденции в технологии разработки ПО и, следовательно, обеспечивает требуемые профессиональные компетенции инженеров-программистов.

СПИСОК ЛИТЕРАТУРЫ

1. Зимняя И.А. Компетентностный подход: каково его место в системе современных подходов к проблеме образования? (Теоретико-методологический аспект)//Высшее образование сегодня. – 2006. – № 8.

2. Болотов В.А., Сериков В.В. Компетентностная модель: от идеи к образовательной программе//Педагогика. – 2003. – № 10.

3. Ларман Крэг. Применение UML 2.0 и шаблонов проектирования: Практическое руководство. – М.: ООО «И.Д.Вильямс», 2009. – 736 с.