ПРОЕКТИРОВАНИЕ ПРОГРАММНЫХ СИСТЕМ: ДИСТАНЦИОННОЕ ОБУЧЕНИЕ

SOFTWARE ENGINEERING: DISTANCE LEARNING

Лукин Владимир Николаевич / Lukin V. N.,
профессор ГОУ ВПО «Московский городской психолого-педагогический университет»/
professor, Moscow State University of Psychology & Education,
lukinvn@list.ru

Сидоров Святослав Игоревич / Sidorov S. I.,
программист ФГАУ «Государственный научно-исследовательский институт информационных технологий и телекоммуникаций» («Информика») /
programmer, State Institute of Information Technologies and Telecommunications,
sis@informika.ru

Фомин Сергей Сергеевич / Fomin S. S.,
начальник отдела повышения квалификации и профессиональной переподготовки кадров ФГАУ «Государственный научно-исследовательский институт информационных технологий и телекоммуникаций» («Информика»)/
head of department, State Institute of Information Technologies and Telecommunications,
fss@informika.ru

Чернышов Лев Николаевич / Chernyshov L. N.,
профессор ФГОБУ ВПО «Государственный университет Министерства финансов Российской Федерации» /
professor, The State University of the Ministry of Finance of the Russian Federation,
levchern@gmail.com

Аннотация

В статье предлагается и обосновывается образовательная программа обучения информационным технологиям в дистанционной форме. Программа представляет собой набор взаимосвязанных дисциплин, посвященных вопросам проектирования программных систем и организации их разработки. Дисциплины программы могут

быть рекомендованы для повышения квалификации и переподготовки специалистов, а также в качестве основы базовой подготовки магистров. Для поддержки дистанционной формы обучения применяется система управления обучением Moodle и специальный инструментарий для организации виртуального практикума.

In this article the educational program on information technology is proposed and justified. The program is designed for distance learning and represents a set of interconnected disciplines devoted to software engineering and organization of its development. The program is recommended for experts training and retraining and as a basis of graduate students training. Distance learning mode is provided with LMS Moodle and the special virtual workshop toolkit.

Ключевые слова: проектирование программных систем, дистанционное обучение, магистратура, виртуальный дистанционный практикум.

Keywords: software engineering, distance learning, graduate students training, distance virtual workshop.

1. Обоснование образовательной программы

Традиционное обучение специалистов, способных выполнять все работы по созданию программных систем от этапа анализа требований до этапа передачи в эксплуатацию и дальнейшего сопровождения, не способствуют насыщению рынка труда квалифицированными кадрами. Возникает стойкое убеждение, что кто-то кого-то обманывает. А именно, вузовская подготовка не приводит и не может привести к получению полноценных знаний, умений и навыков для профессионального программирования, несмотря на обилие разнообразных учебных программ. Косвенно это подтверждает и А.Н. Терехов [1], который прикладывает немало усилий для исправления сложившейся ситуации.

Причина достаточно проста для понимания и весьма сложна для исправления. Многолетняя традиция преподавания программирования в вузах базируется на численных методах – дисциплине строгой и понятной, что провоцирует попытку обучения именно на её основе. Но, как показал Н.Н. Непейвода [2], эта дисциплина далека от процессов в реальной жизни, которые, собственно, и нуждаются в специалистах. Второе, столь любимое вузами, направление – подготовка «олимпиадников», способных решать изощрённые задачи при заданных ограничениях. К реальному программированию это от-

ношения практически не имеет, но у вуза создаётся впечатление, что он готовит классных специалистов. Те же дисциплины, которые дают необходимые знания и умения, обычно идут на старших курсах в рамках спецкурсов. В целом они, как правило, не покрывают всех необходимых квалификационных требований к полноценным специалистам в области прикладного программирования. Кроме того, преподают эти дисциплины далеко не всегда профессиональные программисты, более того, порой преподаватели сами не участвовали в сколь-нибудь значимом программном проекте.

Ситуация осложняется ещё и тем, что скорость изменения информационных технологий создает потребность в эволюционирующих программных продуктах, но пользовательские ожидания и конкурентная борьба существенно затрудняют выпуск качественного программного обеспечения в приемлемые сроки [4]. В этих условиях возникает настоятельная потребность каким-то образом подготовить специалистов, способных осмыслить задачу, оценить требуемые для неё ресурсы, построить адекватную модель, спроектировать систему и подготовить команду, способную этот проект реализовать. Кандидатами на подготовку могут быть специалисты, работающие в области разработки программ, но не имеющие требуемой квалификации, специалисты с техническим образованием, которые предполагают заняться прикладным программированием, выпускники вузов, желающие восполнить пробелы в образовании. Изучение подобных вопросов полезно и при повышении квалификации (переподготовке) преподавателей вузов и колледжей. Неплохо было бы рассмотреть их и в программах для подготовки магистров.

При решении проблемы подготовки кадров всегда встаёт вопрос времени, которое требуется как слушателю, так и преподавателю. Так как в роли слушателя предполагается, вообще говоря, работающий человек, а преподаватель должен иметь практический опыт работы в производстве программных продуктов, эта проблема становится весьма острой. Единственным реальным выходом может служить дистанционное обучение. Но тут надо отметить, что некоторые вопросы подготовки далеко не так просто можно решить с использованием этого подхода. В частности, такой важный вопрос, как формирование проектной команды и руководство ею, требует специальной технологии обучения. Тем не менее, можно сформировать комплекс дисциплин, который, с одной стороны, компенсирует недостатки базового образования, с другой – потребует относительно небольшого времени на изучение, с учетом его дистанционной формы.

Итак, цель образовательной программы подготовки (переподготовки) специалистов – изучение процесса производства программного обеспечения и получение элементарных навыков в этой области, за исключением собственно программирования.

Основное внимание должно уделяться проблемам промышленного производства программных систем среднего масштаба как самых востребованных на рынке. В состав программы должны входить дисциплины, направленные на построение моделей предметной области, архитектуры программных комплексов и взаимодействия их компонентов, а также на изучение средств их производства, включая технологические вопросы. Эта цель хорошо согласуется с требованиями, изложенными в [4].

2. Взгляд сверху: ограничение задачи и покрывающие дисциплины

Для определения круга учебных дисциплин, изучение которых необходимо специалисту в области информационных технологий, рассмотрим обобщённую схему жизненного цикла программной системы (рис. 1). Каждому этапу сопоставим множество наиболее важных для него дисциплин. Сознательно не будем пользоваться термином «программная инженерия», так как его расширительное толкование включает практически все дисциплины, относящиеся к процессу производства программного обеспечения, а ограниченное в значительной степени отражает взгляды и предпочтения толкователя.

Заметим, что дисциплины имеют существенную взаимную зависимость, зачастую их просто трудно разделить [3]. Поэтому определение их содержания неоднозначно, одни дисциплины могут ссылаться на другие, причём, зависимость может быть взаимной. В этом случае зависимая дисциплина, которая читается раньше, должна содержать хотя бы краткое изложение понятий, изучаемых более подробно в следующих дисциплинах.

Приведем список дисциплин с кратким содержанием и указанием на зависимость между ними. Этот комплекс будем называть образовательной программой «Мастер информационных технологий» (МИТ). На рис. 2 представлена взаимосвязь дисциплин МИТ.

1). Технологические основы проектирования программных систем

Общепринятые технологии разработки программного обеспечения, применяемые на этапах анализа требования, проектирования и реализации программных систем (*SADT*, *UML*). Полученные знания полезны, а зачастую и необходимы, при разработке любых программных изделий, независимо от прикладной области и инструментальных средств.

Не зависит от других дисциплин.

2). Методологии проектирования программных систем

Новые и классические подходы в области построения программных систем, эффективность организации труда разработчиков качественных программных продуктов. Методологии, определяющие структуру, логическую организацию деятельности по разработке программных систем, а также методы и средства деятельности. Описание и анализ наиболее распространенных методологий.

Зависит от (1).

3). Психология программирования

Решение психологических проблем при создании работоспособного коллектива разработчиков программного обеспечения и поддержании его устойчивого состояния. Сохранение творческого потенциала коллектива на протяжении текущей разработки и возможности его использования в других проектах.

Не зависит от других дисциплин.

4). Управление разработкой программных систем

Вопросы, связанные с управлением программными проектами: особенности разработки программных проектов, методы планирования работ, стили и способы руководства процессом реализации проектов. Оценка рисков, обоснование выбора модели технологии и разработки. Методы управления, способствующие успешному завершению проекта в заданные сроки при заданных ресурсах. Методы управления бюджетом, мониторинга проекта и соблюдения расписания реализации проекта.

Зависит от (3, 2).

5). Управление требованиями

Анализ и формирование требований к программным системам. Задачи выявления, разработки и документирования требований. *CASE*-средства, поддерживающие этот вид деятельности. Решение проблем, входящих в круг интересов бизнес-аналитиков и менеджеров программных проектов.

Зависит от (2).

6). Экономика программирования

Определение рентабельности разработки и дальнейшей эксплуатации программной системы на основе оценки стоимости, времени и необходимых ресурсов разработки. Характеристики и условия применения моделей оценки. Методики оценки экономической эффективности программного продукта.

Зависит от (2).

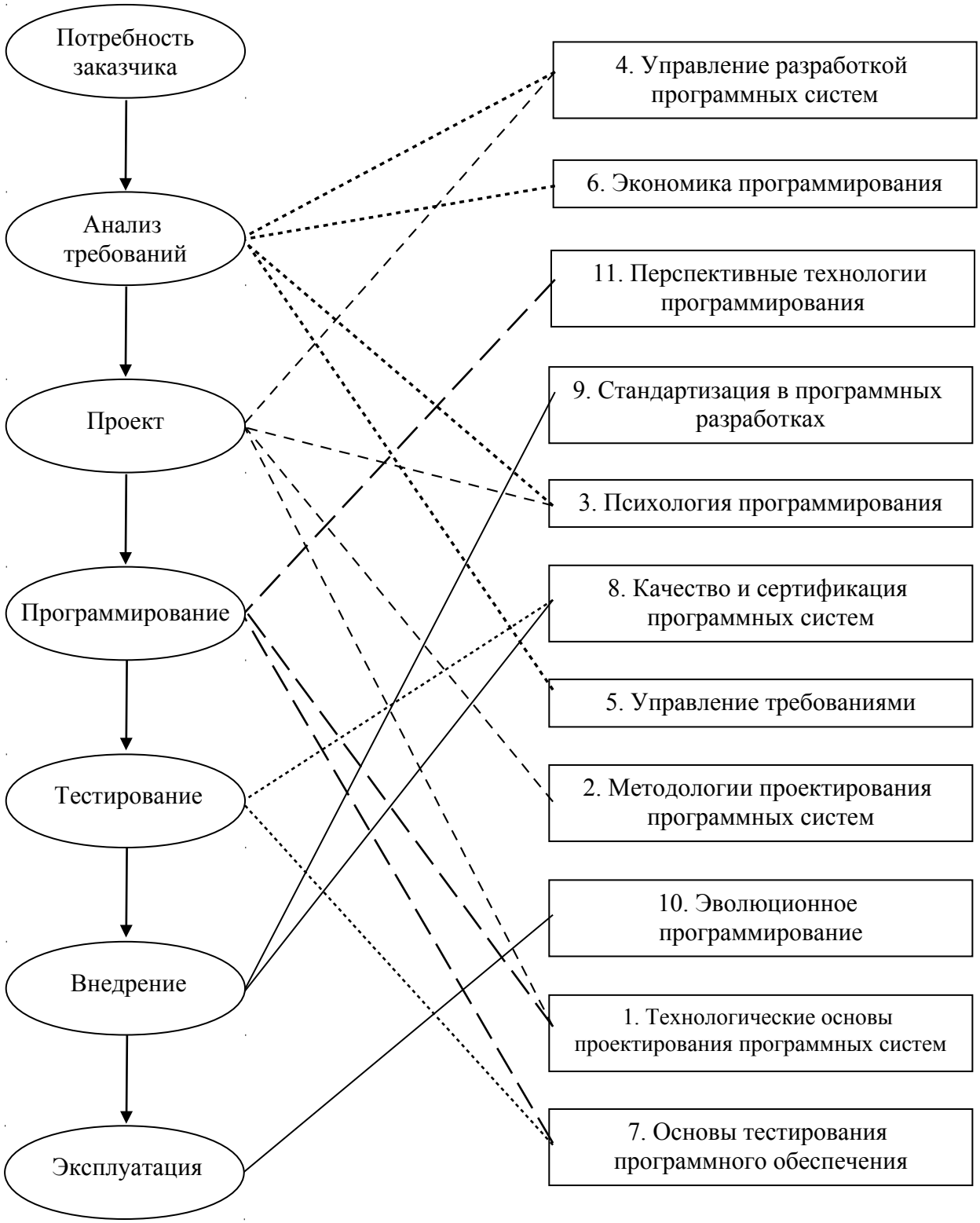

Рис.1. Этапы жизненного цикла программной системы и соответствующие им дисциплины

7). Основы тестирования программного обеспечения

Методики тестирования программ и программных комплексов. Функциональное и структурное тестирование. Тестирование требований, тестирование качества. Системы учета ошибок и анализ надёжности. Инструментальные средства тестирования.

Зависит от (8, 2).

8). Качество и сертификация программных систем

Критерии качества программных систем, проблемы обеспечения и контроля качества. Тестирование качества, стандарты качества. Методы и модели анализа качества информационных систем. Аккредитация и сертификация программных систем.

Зависит от (5, 2, 7).

9). Стандартизация в программных разработках

Ключевые положения стандартов, применяемых при разработке программных систем. Стандарты жизненного цикла, управления разработкой, структуры технической документации, качества программных систем, а также модели определения уровня зрелости коллектива.

Зависит от (8, 2, 1).

10). Эволюционное программирование

Требования к эволюционирующим программным системам. Назначение и содержание процесса сопровождения прикладных программных систем. Анализ и преобразование кода (рефакторинг). Подходы к реорганизации технологии предметной области для существенного повышения эффективности на основе информационных технологий (реинжиниринг бизнес-процессов).

Зависит от (8, 5, 2, 7).

11). Перспективные технологии программирования

Современные перспективные технологии, использование которых способствует повышению производительности труда программистов. Их особенности и области применения. Характеристики программных систем, построенных с применением этих технологий.

Зависит от (1).

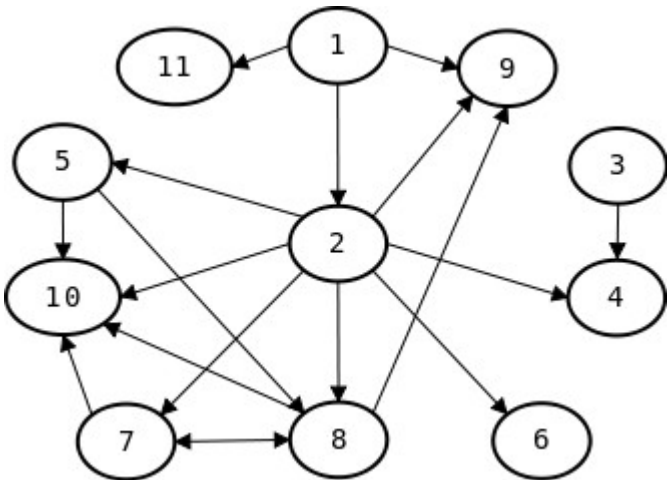

Рис. 2. Взаимосвязь дисциплин МИТ (по номерам в списке)

3. Базовое ядро

Получившееся множество дисциплин, вообще говоря, может удовлетворить запросы слушателя, но оно достаточно трудоёмко для изучения и не всегда всем нужно в полном объеме. Поэтому мы посчитали возможным выделить из него наиболее существенные дисциплины, которые в вузе обычно даются в недостаточном объёме. Они относятся к начальным этапам жизненного цикла, до этапа программирования. Выбор подтверждается мнением, изложенным в [4]: «Из-за сложности программного обеспечения и присущих ему трудностей разработки большинство «сюрпризов», которые происходят в процессе системной интеграции и эксплуатации системы, могут быть отнесены к неправильной реализации и интеграции программного обеспечения, которые стали следствием плохих требований или некорректного обновления программного обеспечения… Роберт Гласс, Нэнси Левенсон и другие приводят многочисленные документально подтвержденные примеры неудач в результате плохого проектирования и/или плохой взаимосвязи между системами и программными инженерами». То есть, наиболее полезно остановиться на таких дисциплинах, как «Управление разработкой программных систем», «Экономика программирования», «Психология программирования», «Управление требованиями», «Методологии проектирования программных систем», «Технологические основы проектирования программных систем». Это множество дисциплин стоит разбить на два: в первом будут независимые дисциплины, во втором – зависимые. Тогда в первом окажутся дисциплины «Психология программирования» и «Технологические основы проектирования программных систем». Учитывая, что методологии разработки очень тесно связаны с технологиями, порой до полного смешения терминов, их следует рассматривать в связке.

Резюмируя, получаем следующее. В первую очередь, именно в таком порядке, следует изучать дисциплины, которые можно назвать базовым ядром: «Технологические основы проектирования программных систем», «Методологии проектирования программных систем», «Психология программирования».

Второй цикл обучения составляют дисциплины «Управление разработкой программных систем», «Управление требованиями», «Экономика программирования».

И, наконец, менее существенными, хотя и полезными, представляются «Основы тестирования программного обеспечения», «Качество и сертификация программных систем», «Стандартизация в программных разработках», «Эволюционное программирование», «Перспективные технологии программирования».

4. Методология преподавания

Объем каждой дисциплин в часах определяется ее содержанием, но не должен быть менее 72 часов, которые требуются для получения сертификата о повышении квалификации. Календарная длительность изучения дисциплины определяется из расчета времени, которое слушатель реально может выделить. Считаем, что в среднем она не может превышать двух часов в день. График работы в условиях дистанционного обучения относительно свободный. Таким образом, на одну дисциплину отводится 36 рабочих дней, около 2-х месяцев. Одновременно слушатель может проходить две дисциплины, если это не противоречит их взаимозависимости.

Если исходить из минимального объёма, базовая программа из трех дисциплин укладывается в 6 месяцев при последовательном изучении. Весь цикл из 11 дисциплин потребует двух лет обучения (с учетом отпусков). Заметим, этот срок соответствует двухлетней магистерской подготовке.

Таблица 1

Объем часов по дисциплинам

№ п/п	Наименование дисциплины	Всего часов	В том числе			
			Изучение теоретического материала	Самостоятельная работа	Практические занятия	Тестирование знаний
1	Технологические основы проектирования программных систем	72	25	18	22	7
2	Методологии проектирования программных	72	25	25	12	10

	систем					
3	Психология программирования.	72	25	27	10	10
4	Управление разработкой программных систем	72	25	20	20	7
5	Управление требованиями	72	25	18	22	7
6	Экономика программирования	72	25	20	20	7
7	Основы тестирования программного обеспечения	72	25	15	25	7
8	Качество и сертификация программных систем	72	35	10	20	7
9	Стандартизация в программных разработках	72	35	20	10	7
10	Эволюционное программирование	72	20	20	25	7
11	Перспективные технологии программирования	72	35	20	10	7
	Итого:	*792*	*300*	*213*	*196*	*83*

Несмотря на то, что образовательная программа МИТ задумывалась как единое целое, слушатель, в зависимости от своей подготовки и желания, может выбрать для изучения как отдельные дисциплины, так и их наборы. Приведём примеры цепочек дисциплин, соответствующих наименованиям специальностей из перечня, определенного АПКИТ [7]. Порядок изучения дисциплин приблизительно соответствует их взаимозависимости, приведённой на рис. 2.

Руководитель программного проекта, менеджер информационных технологий

- «Методологии проектирования программных систем»,
- «Психология программирования»,
- «Управление разработкой программных систем»,
- «Экономика программирования»,
- «Перспективные технологии программирования».

Системный аналитик

- «Технологические основы проектирования программных систем»,
- «Методологии проектирования программных систем»,
- «Управление требованиями».

Проектировщик

- «Технологические основы проектирования программных систем»,
- «Методологии проектирования программных систем»,
- «Управление требованиями»,
- «Качество и сертификация программных систем»,
- «Основы тестирования программного обеспечения»,
- «Перспективные технологии программирования».

Специалист по информационным системам

- «Технологические основы проектирования программных систем»,
- «Методологии проектирования программных систем»,
- «Качество и сертификация программных систем»,
- «Стандартизация в программных разработках»,
- «Перспективные технологии программирования».

5. Магистратура

При разработке учебных планов полезен зарубежный опыт, который в максимальной степени сконцентрирован в документах, выпускаемых совместным комитетом по образованию сообществ *ACM* и *IEEE Computer Society*. Недавно вышедший документ «*Graduate Software Engineering* 2009 (*GSwE2009*): *Curriculum Guidelines for Graduate Degree Programs in Software Engineering*» содержит рекомендации по магистерской подготовке в области программной инженерии. Ниже приводятся некоторые требования к выпускнику магистерской программы, которые можно считать целями подготовки и предлагаемой образовательной программы.

Магистр программного обеспечения должен

- быть квалифицированным специалистом, по крайней мере, в одной прикладной области, такой, как финансы, медицина, транспорт или телекоммуникации, и в одном из типов приложений, например, в приложениях режима реального времени, встроенных приложениях, приложениях, критически важных по безопасности или распределенных системах. Его квалификация включает

понимание специфики прикладной области и особенностей соответствующих ей типов приложений, проявляющихся в программной системе и в инструментарии, а также включает владение методами изучения как прикладных областей, так и типов приложений.

- принимать этически правильные профессиональные решения и практиковать этическое профессиональное поведение.
- быть эффективным членом команд, в том числе международных и территориально-распределенных, эффективно общаться как в устной, так и в письменной форме, быть лидером в одной из областей разработки проектов, таких, как управление проектами, анализ требований, архитектура, конструирование или обеспечение качества.
- уметь сочетать противоречивые проектные цели, находя приемлемые компромиссы в пределах ограничений стоимости, времени, ограничений существующих систем и организации.
- понимать и оценивать технико-экономические требования, уметь вести переговоры и успешно взаимодействовать со всеми заинтересованными сторонами в типичной среде разработки программного обеспечения, а также быть в состоянии качественно выполнять эти задачи, иметь эффективные трудовые навыки и быть руководителем.
- быть способным изучать новые модели, методы и технологии по мере их появления, осознавать необходимость непрерывного профессионального развития.
- уметь анализировать текущие технологии программного обеспечения, формулировать их сильные и слабые стороны, сравнивать их с альтернативными, а также способствовать улучшению или расширению этих технологий.

В новых образовательных стандартах, на которые переходит высшее образование в России, требования к выпускникам формулируются в виде компетенций. Однако в реальности пока содержание курсов не всегда им соответствует. Отчасти это обусловлено новой формой учебных программ и тем, что при переходе к государственным образовательным стандартам 3-го поколения у выпускающих кафедр появилась большая свобода по формированию учебных планов, особенно для второго уровня образования – магистерской подготовки. При отсутствии жесткой регламентации по составу и содержанию дисциплин сложно обеспечить единые

высокие требования к выпускникам. Здесь многое зависит от коллективов кафедр, уровня квалификации преподавательского состава. При непрерывно развивающихся информационных технологиях трудно обеспечить учебный процесс по новым направлениям, если в коллективе нет соответствующих специалистов. Привлечение же специалистов из производства не очень распространено. В результате новые технологии просто не будут присутствовать в учебных программах.

В какой-то степени проблему повышения качества подготовки программистов может решить дистанционное обучение. С одной стороны, слушатели, которые решились на такой способ обучения, безусловно, более мотивированы, чем студенты дневных отделений. С другой стороны, имеется возможность привлечь к преподаванию квалифицированных преподавателей и работающих специалистов с большим опытом.

Предлагаемая образовательная программа может рассматриваться как своего рода сокращённый вариант магистерской подготовки по специальности программной инженерии. Программа МИТ имеет меньший объем часов за счет отсутствия дисциплин фундаментальной направленности: её дисциплины ориентированы на определенную область деятельности. В рамках программы МИТ возможен выбор цепочек дисциплин, что согласуется с предоставлением Государственными образовательными стандартами третьего поколения по магистерским программам большой свободы выбора.

6. Технологические основы проектирования программных систем

В качестве примера приведём сведения о программе дисциплины, опытная эксплуатация которой была проведена во ФГАУ «Государственный научно-исследовательский институт информационных технологий и телекоммуникаций» (Информика) совместно с Государственным университетом Министерства финансов РФ в 2011 году. Это «Технологические основы проектирования программных систем».

Технологию разработки программного обеспечения нередко определяют как программную инженерию – технологию, основанную на систематическом подходе к анализу, проектированию, реализации и поддержке программного обеспечения, которая базируется на использовании разнообразных моделей жизненного цикла и методологий, связанных с различными его стадиями, и во многих случаях предусматривает использование разнообразных *CASE*-инструментов. Здесь мы будем её понимать именно как технологию: комплекс методов, стандартов и инструментальных средств, используемых для создания, поддержки и применения компьютерных систем какого-либо класса в некоторой среде функционирования.

Таким образом, данная дисциплина входит первой в цепочку дисциплин, ориентированных на проектирование программных систем, на ней базируется дисциплина «Методологии проектирования программных систем».

6.1. Цели и задачи дисциплины

Цель дисциплины – изучение общепринятых технологий разработки программного обеспечения, применяемых на этапах анализа требований, проектирования и реализации программных систем, независимо от прикладной области. Знание технологий разработки позволит в рамках принятой методологии обеспечить производство качественного программного обеспечения.

6.2. Требования к уровню усвоения содержания дисциплины

Слушатель должен

- *иметь представление* о современных взглядах на проблемы технологического обеспечения процессов жизненного цикла программного обеспечения;
- *знать* технологические основы методологий структурного анализа и проектирования на основе семейства *IDEF* и объектно-ориентированного – на основе языка моделей *UML*;
- *уметь* применять правила построения моделей и диаграмм в соответствии с методологиями семейства *IDEF* и методологиями, основанными на языке *UML*.

Таблица 2

Объем дисциплины и виды учебной деятельности

Виды учебной работы	Всего часов
Общая трудоемкость дисциплины	72
Занятия в дистанционной форме	
Изучение теоретического материала	25
Выполнение практических заданий	22
Самостоятельная работа	18
Тестирование знаний	7

6.3. Содержание дисциплины

Дисциплина включает структурные и объектно-ориентированные технологии разработки программного обеспечения, применяемые на различных этапах жизненного цикла программных систем. Рассматриваются такие технологии, как *SADT* и *UML*, приводятся рекомендации по их использованию.

Таблица 3

Разделы дисциплины и виды занятий

	Занятия в дистанционной форме				
Наименование темы	Изучение теоретического материала	Выполнение практических заданий	Самостоятельная работа	Тестирование знаний	Всего часов
Тема 1. Технологии структурного анализа требований	4	4	4	1	13
Тема 2. Технологии структурного проектирования	6	6	4	1	17
Тема 3. Объектно-ориентированное проектирование с использованием языка моделирования *UML*	2	2	2	1	7
Тема 4. Диаграммы вариантов использования и классов в *UML*	4	4	2	1	11
Тема 5. Диаграммы деятельности в *UML*	3	2	2	1	8
Тема 6. Диаграммы кооперации и последовательности в *UML*	3	2	2	1	8
Тема 7. Диаграммы состояний в *UML*	3	2	2	1	8
ИТОГО:	25	22	18	7	**72**

6.4. Содержание разделов дисциплины

Тема 1. Технологии структурного анализа требований

Основные понятия и принципы структурного анализа. Методология *SADT*, область её применимости. Особенности и правила методологии *SADT*. Примитивы графической модели, виды функциональных связей. Отношения, реализуемые функциональными связями. Этапы построения диаграмм *IDEF0*. Контекстные диаграммы, детализация. Целостность модели.

Тема 2. Технологии структурного проектирования

Семейство методологий *IDEF*. Модель данных *IDEF1X*. Модель потоков данных, построение диаграмм *DFD*. Контекстные диаграммы, детализация. Целостность модели. Достоинства и недостатки структурных методологий.

Тема 3. Объектно-ориентированное проектирование с использованием языка моделирования UML

Объектно-ориентированный подход к моделированию программных систем. Компоненты универсального языка моделирования (*UML*). Диаграммы, правила их построения. Статические и динамические диаграммы.

Тема 4. Диаграммы вариантов использования и классов в UML

Диаграммы вариантов использования, их применимость и правила их построения. Диаграммы классов – основной тип диаграмм в методологиях, основанных на *UML*. Возможности и применение диаграмм классов.

Тема 5. Диаграммы деятельности в UML

Конечные автоматы. Модель состояний и событий и её реализация конечными автоматами. Реализация конечных автоматов диаграммами деятельности: основные компоненты, правила построения.

Тема 6. Диаграммы кооперации и последовательности в UML

Назначение диаграмм кооперации. Сообщения. Синхронные и асинхронные потоки. Правила построения диаграмм кооперации. Назначение диаграмм последовательности. Взаимодействие процессов. Правила построения диаграмм последовательности.

Тема 7. Диаграммы состояний в UML

Реализация конечных автоматов диаграммами состояний: основные компоненты, правила построения. Реализация иерархии диаграмм состояний.

Таблица 4

Понедельный план проведения лекционных и практических занятий

Неделя	**Разделы и темы дисциплины**	**Изучение теоретического материала (час.)**	**ПЗ (час.)**
1	Технологии структурного анализа требований	4	4
1-3	Технологии структурного проектирования	6	6

3	Объектно-ориентированное проектирование с использованием языка *UML*	2	2
3-4	Диаграммы вариантов использования и классов в *UML*	4	4
4-5	Диаграммы деятельности в *UML*	3	2
5	Диаграммы кооперации и последовательности в *UML*	3	2
6	Диаграммы состояний в *UML*	3	2

Трудоёмкость изучения дисциплины в дистанционном режиме составляет 10 часов в неделю (2 часа в день).

Таблица 5

Практикум

№ темы	Название темы	Наименование практического занятия
1	Технологии структурного анализа требований	Изучение средства *BPwin* (*AllFusion Process Modeler*). Построение диаграмм *IDEF0* с использованием *BPwin*.
2	Технологии структурного проектирования	Изучение средства *ERwin*. Построение модели *IDEF1X*. Построение *DFD*-диаграмм.
3	Объектно-ориентированное проектирование с использованием языка *UML*	Основы работы в среде объектно-ориентированного *CASE*-средства
4	Диаграммы вариантов использования и классов в *UML*	Диаграммы вариантов использования и диаграммы классов
5	Диаграммы деятельности в *UML*	Составление диаграмм деятельности
6	Диаграммы кооперации и последовательности в *UML*	Составление диаграмм кооперации и последовательности для учебного проекта
7	Диаграммы состояний в *UML*	Составление диаграмм состояний

Как показала опытная эксплуатация, слушатели положительно восприняли материал и способ его подачи. Теоретического материала было достаточно для понимания сути дисциплины и выполнения практических работ. Практические работы дали возможность получить элементарные навыки работы с инструментальными средствами при анализе и проектировании учебных программных систем. Дистанционный метод обучения, разумеется, на первых порах вызывал определённые трудности, но в целом оказался адекватным.

7. Реализация

Дисциплины программы МИТ рассчитаны на дистанционное обучение и являются практико-ориентированными: слушателю не требуется приезжать на занятия в

обучающую организацию и устанавливать на свой компьютер дополнительное программное обеспечение. Это достигается за счет использования системы управления обучением *Moodle*, в которой публикуется учебный материал, включая тесты (рис. 3), а также использованием так называемых «виртуальных сред», к которым слушатель получает доступ через интернет для выполнения практических заданий.

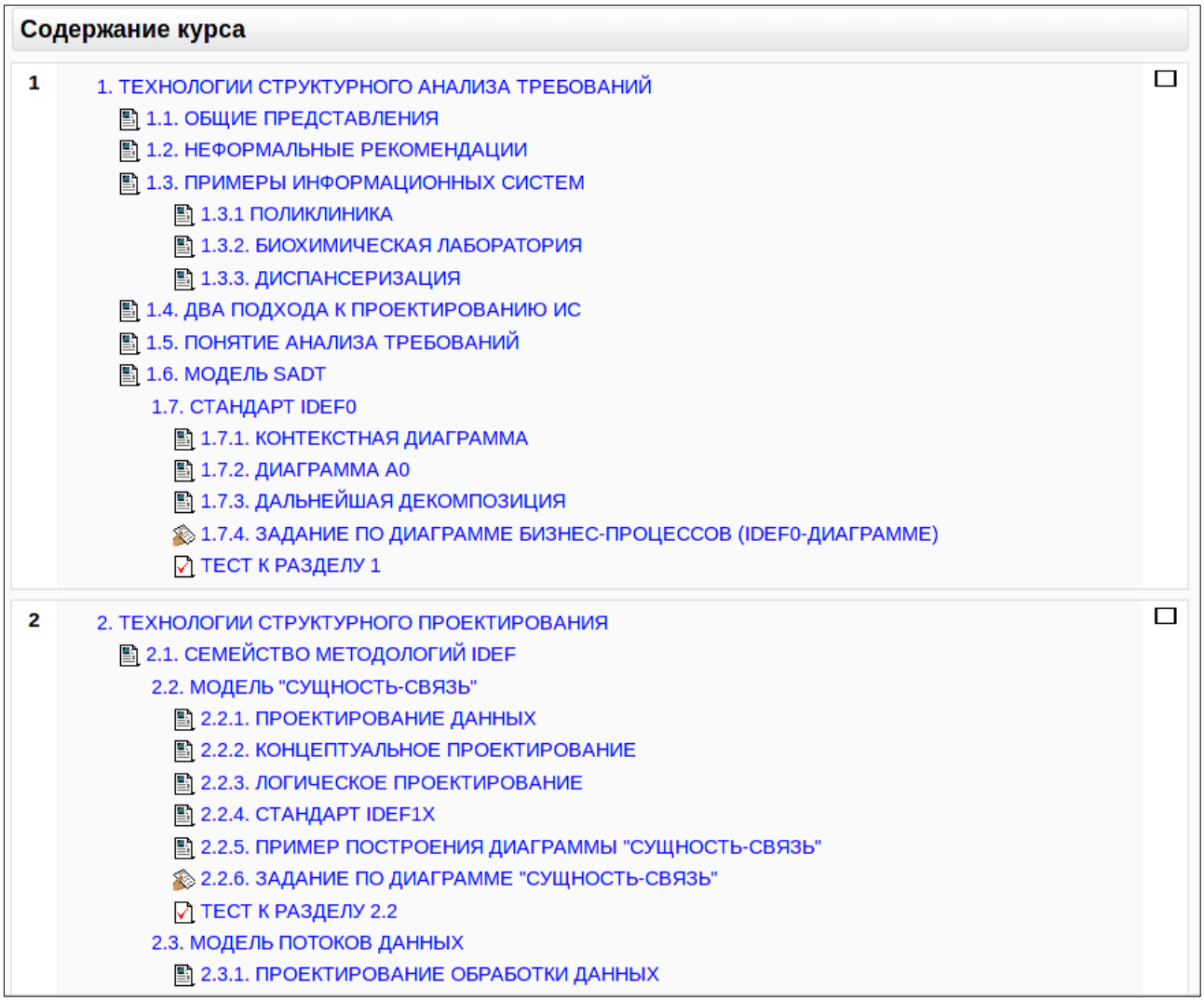

Рис. 3. Размещение учебного материала в системе управления обучением *Moodle* для организации дистанционного обучения

Виртуальные среды реализованы на базе виртуальных машин и размещены на серверах Информики. В виртуальной среде слушателя подготовлено программное обеспечение, необходимое для выполнения задания. Здесь же слушатель сохраняет результаты работы, доступные для проверки сетевому преподавателю. В каждом практическом задании слушателю доступна ссылка на страницу управления его виртуальной средой (ссылка «Соединение с виртуальным практикумом», рис. 4).

1.7.4. Задание по диаграмме бизнес-процессов (IDEF0-диаграмме)

Соединение с виртуальным практикумом

Задача 1
Создайте модель биохимической лаборатории поликлиники, включающую контекстную диаграмму с единственным блоком без стрелок. Если кегль или гарнитура шрифта неудобны для восприятия, измените их. Задайте определение блока и примечание к нему.

Задача 2
Определите входные потоки модели и задайте их на контекстной диаграмме. Завершите работу с сохранением модели.

Рис. 4. Ссылка «Соединение с виртуальным практикумом»: перейдя по ней, слушатель попадает на страницу управления своей виртуальной средой

Страница управления виртуальной средой слушателя («Виртуальный практикум») показана на рис. 5.

Виртуальный практикум
1.7.4. ЗАДАНИЕ ПО ДИАГРАММЕ БИЗНЕС-ПРОЦЕССОВ (IDEF0-ДИАГРАММЕ)

Статус виртуальной среды: Выключена

Создать ВС | Статус ВС
Удалить ВС | Параметры соединения
Скачать результат | Отправить на проверку

Кнопки управления виртуальной средой:

"Создать ВС"	- создание индивидуальной виртуальной среды (ВС) для выполнения в ней задания
"Удалить ВС"	- уничтожение ВС с сохранением результатов работы на сервере обучающей организации
"Скачать результат"	- скачать результат выполнения задания в виде zip-архива
"Статус ВС"	- вывод текущего статуса ВС
"Параметры соединения"	- вывод данных для соединения с ВС
"Отправить на проверку"	- отправка уведомления преподавателю о

Рис. 5. Страница управления виртуальной средой (виртуальный практикум)

Чтобы начать выполнение практического задания, слушатель нажимает кнопку «Создать ВС». В результате для него будет создана виртуальная среда. Ее статус после создания – «Включена». С ней можно установить соединение, используя «Клиент удаленного рабочего стола» и параметры, появившиеся в окне внизу страницы (рис. 6.).

Данные для входа в Вашу виртуальную среду:	
Логин в ВС	student1
Пароль в ВС	eCs7Nkp8
Адрес доступа	xxxxxx.informika.ru
Порт доступа	50200

Рис. 6. Параметры соединения с виртуальной средой

Введя адрес и порт доступа в соответствующие поля «Клиента удаленного рабочего стола», слушатель получает доступ к своей виртуальной среде. В ней он аутентифицируется, используя предоставленные логин и пароль. После этого слушатель может приступить к выполнению практического задания с использованием установленного в виртуальной среде программного обеспечения (рис. 7).

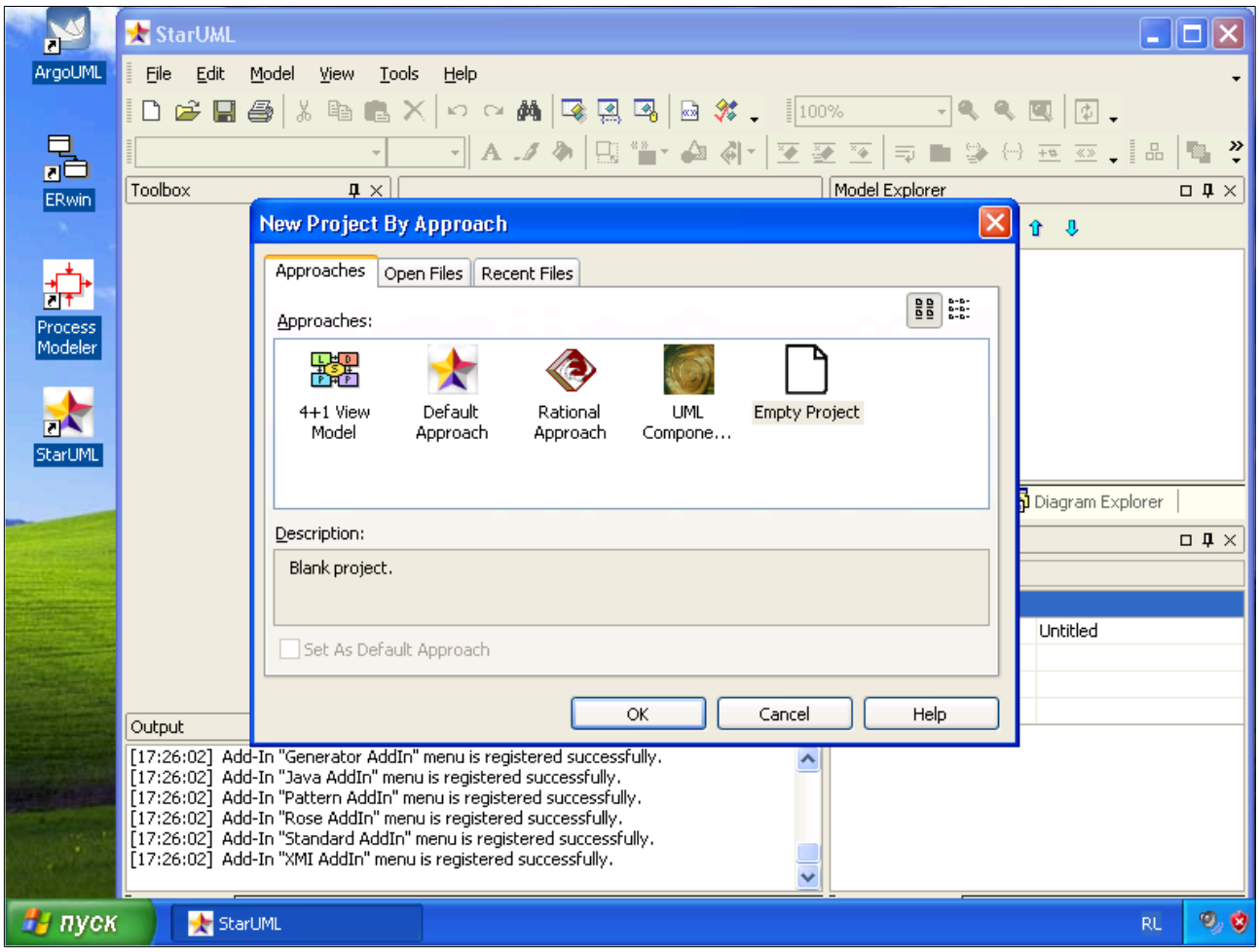

Рис. 7. Выполнение практического задания в виртуальной среде

Выполнив задание, слушатель сохраняет результат в виртуальной среде, используя для этого указанный в инструкции каталог, после чего выходит из виртуальной среды («Пуск» → «Выход из системы») и нажимает в виртуальном практикуме кнопку «Отправить на проверку». Таким образом, слушатель предоставляет сетевому преподавателю доступ к результатам своей работы и уведомляет его о необходимости их проверки. На время проверки работы слушатель не имеет доступа к виртуальной среде и, таким образом, не может помешать сетевому преподавателю.

Сетевой преподаватель получает по электронной почте уведомление о необходимости проверки задания, выполненного слушателем. В автоматически сгенерированном письме указываются номер задания, ФИО слушателя, а также параметры соединения с его виртуальной средой. Сетевой преподаватель входит в виртуальную среду слушателя, используя «Клиент удаленного рабочего стола», и проверяет его работу. Если качество выполненной работы удовлетворяет сетевого преподавателя, он выставляет слушателю оценку. Если же оно неудовлетворительно, сетевой преподаватель высылает слушателю уведомление о необходимости доработки полученных результатов. Эти возможности реализованы в интерфейсе виртуального

практикума для сетевого преподавателя, который отличается от интерфейса виртуального практикума для слушателя элементами, показанными на рис. 8 и 9.

Выберите группу: Тест Показать

Вывод списка слушателей, выполнивших задание

Обновить информацию

ВС слушателей из группы Тест

Логин	Статус ВС	Оценка	Время создания ВС
student1	вкл	нет	Четверг 26 Январь 2012, 16:11

Рис. 8. Таблица слушателей, выполнивших данное задание

В интерфейсе виртуального практикума для сетевого преподавателя имеется таблица слушателей, выполнивших данное задание (тех, которые сдали задание на проверку, а также тех, которые уже получили за него оценку). В меню «Выберите группу» сетевой преподаватель выбирает ту группу, с которой он работает. Логин каждого слушателя в таблице является ссылкой, при нажатии на которую в текстовое окно внизу страницы выводятся параметры соединения с его виртуальной средой и некоторые другие данные (рис. 9).

Детальная информация по данному слушателю:

Имя	Логин в ВС	Пароль в ВС	Адрес (RDP)	Время создания ВС
Тестовый Пользователь	student1	XTcXz5T5	xxxxxx.informika.ru:50200	Четверг 26 Январь 2012, 16:11

Отправить на доработку Оценить Скачать результат

Рис. 9. Детальная информация о виртуальной среде выбранного слушателя

Под таблицей на рис. 9 представлено несколько ссылок. Ссылка «Отправить на доработку» позволяет средствами виртуального практикума отправить слушателю уведомление о необходимости доработки задания. Такое уведомление будет включать замечания сетевого преподавателя, вводимые при отправке. По ссылке «Оценить» открывается окно *Moodle* для выставления оценки за выполнение практического задания. Введенная оценка сохраняется в стандартной зачетной книжке *Moodle*. Ссылка «Скачать результат» позволяет сетевому преподавателю скачать работу слушателя на домашний компьютер в случае необходимости. После выставления оценки виртуальная среда слушателя будет удалена автоматически, а ее статус для слушателя в соответствующем задании будет изменен на «Выключена». Так как все работы слушателей автоматически сохраняются на сервере обучающей организации, есть возможность скачать их и после того, как практическое задание сдано, а соответствующая виртуальная среда удалена.

Итак, выполнение слушателем практического задания состоит из следующих шагов:

- слушатель читает условие задания, проходит в виртуальный практикум и создает виртуальную среду;
- слушатель соединяется с виртуальной средой и выполняет в ней практическое задание;
- выполнив задание, слушатель сохраняет результат в виртуальной среде, закрывает соединение с ней и отправляет сетевому преподавателю уведомление о необходимости проверки работы;

- сетевой преподаватель, получив уведомление, заходит в удобное для него время в виртуальную среду слушателя и проверяет его работу; если качество работы удовлетворительно, сетевой преподаватель выставляет слушателю оценку, иначе он отправляет слушателю уведомление о необходимости доработки результатов;
- в случае необходимости, слушатель дорабатывает результаты и вновь высылает сетевому преподавателю уведомление о проверке; доработка задания может производиться неоднократно, пока сетевой преподаватель не сочтет возможным выставить слушателю оценку.

Предлагаемая реализация учебного процесса по дисциплинам МИТ предоставляет и дополнительные возможности:

- все слушатели еженедельно получают уведомление о задолженностях по практическим заданиям и тестам, которое генерируется и рассылается автоматически;
- все сетевые преподаватели еженедельно получают отчет об успеваемости закреплённых за ними слушателей, который генерируется и рассылается автоматически;
- все сетевые преподаватели ежедневно получают отчет о заданиях, которые необходимо проверить в текущую дату. Данный отчет также генерируется и рассылается автоматически;
- слушатель может скачать на домашний компьютер результаты своей работы (кнопка «Скачать результат»), узнать текущий статус виртуальной среды и параметры соединения с ней (кнопки «Статус ВС» и «Параметры соединения», соответственно) (рис. 5);
- слушатель может удалить свою виртуальную среду (кнопка «Удалить ВС») в том случае, если он нарушил работу программного обеспечения в виртуальной среде и уже не может полноценно выполнять практическое задание. Удалив виртуальную среду, слушатель нажимает кнопку «Создать ВС» и получает новую исправную виртуальную среду;
- если слушатель, сохранив результаты работы и отправив уведомление сетевому преподавателю, не закрыл соединение с виртуальной средой, это соединение будет закрыто автоматически;
- сетевой преподаватель имеет возможность поработать в собственной виртуальной среде по каждому практическому заданию, опробовать установленное там программное обеспечение и приемы работы с ним: в

интерфейсе виртуального практикума для сетевого преподавателя присутствуют те же кнопки управления виртуальной средой, что и в интерфейсе пользователя.

8. Заключение

Предлагаемая образовательная программа повышения квалификации «Мастер информационных технологий» будет полезна для широкого круга слушателей: студентов старших курсов, магистров и специалистов, стремящихся повысить свою квалификацию в области проектирования программных систем. Привлекательные черты программы – практическая ориентированность и дистанционная форма обучения. Они дают возможность получать навыки и умения при минимальных трудозатратах.

Дисциплина «Технологические основы проектирования программных систем» готова к проведению обучения. Подробности на сайте ***proteus.informika.ru****. Заявки на обучение от частных лиц и организаций принимаются по электронной почте* ***courses@informika.ru****.*

Предлагаемая реализация учебного процесса имеет следующие достоинства:

- полная дистанционность обучения, включая выполнение практических заданий и тестов;
- круглосуточный доступ к учебным материалам и виртуальному практикуму;
- отсутствие необходимости установки на домашнем компьютере слушателя и сетевого преподавателя дополнительного программного обеспечения;
- возможность незамедлительно получить новую работоспособную виртуальную среду с исходной конфигурацией в случае выхода из строя программного обеспечения в текущей виртуальной среде;
- сохранность результатов выполнения практических работ слушателей на серверах обучающей организации и возможность скачать эти результаты на домашний компьютер для слушателей и сетевых преподавателей;
- автоматическое администрирование виртуальных сред на серверах обучающей организации;
- автоматическое составление и рассылка уведомлений и отчетов об успеваемости, проверке заданий и задолженностях для сетевых преподавателей и слушателей.

Литература

1. Терехов А.Н. Технология программирования: Учебное пособие. – М.: Интернет-Университет Информационных Технологий; БИНОМ. Лаборатория знаний, 2006.
2. Непейвода Н.Н. Факультет прикладной логики: информатики-аналитики // Седьмая конференция «Свободное программное обеспечение в высшей школе»: Тезисы докладов / Переславль, 28 – 29 января 2012 года. – М.: Альт Линукс, 2012. – С. 13 – 15.
3. Гецци К., Джазайери М., Мандриоли Д. Основы инженерии программного обеспечения. 2-е изд.: Пер. с англ. – СПб.: БХВ-Петербург, 2005.
4. Graduate Software Engineering 2009 (GSwE2009): Curriculum Guidelines for Graduate Degree Programs in Software Engineering. Version 1.0, 2009. Stevens Institute of Technology.
5. Лукин В.Н., Чернышов Л.Н. Опыт использования приложения с открытым кодом StarUML в системе дистанционного обучения MOODLE с виртуальным практикумом // Седьмая конференция «Свободное программное обеспечение в высшей школе»: Тезисы докладов / Переславль, 28 – 29 января 2012 года. – М.: Альт Линукс, 2012. – С. 45 – 46.
6. Кривошеев А.О., Сидоров С.И., Фомин С.С. Организация виртуального дистанционного практикума для системы повышения квалификации в области ИКТ // Информатизация образования и науки. – 2011. – № 4 (12). – С. 3 – 14.
7. Профессиональные стандарты в области ИТ. – М.: АП КИТ, 2008.