Интернет-журнал «Науковедение» ISSN 2223-5167 http://naukovedenie.ru/

Том 8, №5 (2016) http://naukovedenie.ru/index.php?p=vol8-5

URL статьи: http://naukovedenie.ru/PDF/66TVN516.pdf

Статья опубликована 14.11.2016.

Ссылка для цитирования этой статьи:
Романов Е.Л., Трошина Г.В., Якименко А.А. Преподавание программной инженерии. Взгляд от кода // Интернет-журнал «НАУКОВЕДЕНИЕ» Том 8, №5 (2016) http://naukovedenie.ru/PDF/66TVN516.pdf (доступ свободный). Загл. с экрана. Яз. рус., англ.

УДК 378, 004.4

Романов Евгений Леонидович
ФГБОУ ПО «Новосибирский государственный технический университет», Россия, Новосибирск[1]
Кандидат технических наук, доцент
E-mail: romanov@corp.nstu.ru
РИНЦ: http://elibrary.ru/author_items.asp?authorid=251557

Трошина Галина Васильевна
ФГБОУ ПО «Новосибирский государственный технический университет», Россия, Новосибирск
Кандидат технических наук, доцент
E-mail: troshina@corp.nstu.ru
РИНЦ: http://elibrary.ru/author_profile.asp?id=661106

Якименко Александр Александрович
ФГБОУ ПО «Новосибирский государственный технический университет», Россия, Новосибирск
Кандидат технических наук, доцент
E-mail: al--le@yandex.ru
РИНЦ: http://elibrary.ru/author_profile.asp?id=768964

Преподавание программной инженерии. Взгляд от кода

Аннотация. В статье рассматриваются аспекты высшего образования в направлении подготовки «Программная инженерия». Отмечается историческая преемственность с направлением «Информатика и вычислительная техника». Проводятся детальный анализ и критическая оценка профессиональных стандартов в области информационных технологий и образовательных стандартов подготовки бакалавров и магистров с позиций свода знаний программной инженерии. Отмечается, что образовательный стандарт бакалавра в целом согласуется с требованиями стандарта профессии «Программист» и выдержан в русле свода знаний. Проанализированы недостатки образовательного стандарта магистратуры, затрудняющие подготовку по профессиям этого ряда - «Архитектор программного обеспечения» и «Руководитель разработки». Обоснована методология обучения «от кода», основанная на последовательном усложнении практики программирования, перечисляются ее этапы. Обсуждается баланс творческого и рутинного начал в профессии, сложившиеся стереотипы. Показано, что требования, предъявляемые к архитекторам программного обеспечения, не могут быть выполнены без глубоких знаний и практики кодинга и проектирования и находятся в русле направления «Программная инженерия». Применительно к направлению обсуждаются общие «больные места» психологии студентов и организации учебного процесса: учебные задачи vs деловые игры, копипаст, ориентация на готовые решения, уровень абстрагирования, учеба как самоцель, проблема сложности разработки,

[1] 630073, г. Новосибирск, пр-т К. Маркса, 20

неразвитость индуктивного подхода, схоластика образования, фундамент vs практика. Отмечены полезные стороны дисциплины «Программная инженерия» для родственных направлений.

Ключевые слова: программная инженерия; профессиональные и образовательные стандарты; программист; кодинг; практика программирования и проектирования

«Учитель сказал: если хороший человек учил людей семь лет, их можно посылать в сражение»
Конфуций

В 2011 году на кафедре вычислительной техники Новосибирского государственного технического университета (НГТУ) открыто направление подготовки бакалавров и магистров «Программная инженерия» (ПИ). Традиционным для кафедры направлением было и остается «Информатика и вычислительная техника» (ИВТ), в учебном плане которой «Программная инженерия» читается в качестве отдельной дисциплины. И такая ситуация типична. В настоящее время подготовка по направлению «Программная инженерия» ведется в 90 ВУЗах по 101 программе[2,3]. География направления довольно узка: Москва - 12 ВУЗов, 15 программ, Санкт-Петербург – 7(10), Томск – 3(5), Казань – 4(5), Волгоград, Красноярск – 3(3), Саратов - 2(2), Новосибирск – 1(2). Для сравнения по направлению ИВТ – 251 ВУЗ, 304 программы: Москва – 24(54), Санкт-Петербург – 12(26), Новосибирск – 4(6). Весьма поверхностный анализ учебных планов показывает их явную преемственность с направлением ИВТ. Отличие состоит в появлении дисциплин, связанных с компонентами жизненного цикла программного обеспечения (ПО) – управление требованиями, проектирование, тестирование, управление программными проектами, квалиметрия и т.п.

От стандарта до реальности

Основной рамочной конструкцией учебного процесса являются стандарты: профессиональные[4] и образовательные[5]. Кроме того, известны общепризнанные рамочные нормативы в виде «Свода знаний о программной инженерии» (SWEBOK), включающие подробные рекомендации по структуре направления [1] и методике преподавания [2]. Вопрос в том, что в реальности полностью совместить эти документы в структуре образовательного процесса бывает сложно. Для этого дадим субъективную оценку перечисленным документам.

Свод знаний о программной инженерии (SWEBOK) [3] содержит самое общее согласованное описание областей знаний и артефактов программной инженерии. В этом смысле он является отправной точкой для привязки всех последующих документов. Большинство областей знаний идентично технологическим процессам (дисциплинам, деятельностям) методологии унифицированного процесса (UP) разработки ПО. В то же время

[2] Специальность (09.03.04). Программная инженерия в вузах России / ПроВУЗ.ру [Электронный ресурс]: режим доступа – http://www.provuz.ru/vuz/spec/fgos090304/.

[3] Программная инженерия — бакалавриат (09.03.04) / Вузы России [Электронный ресурс]: режим доступа – http://vuz.edunetwork.ru/specs/122 (дата обращения: 1.7.2016).

[4] Профессиональные стандарты / Министерство труда и социальной защиты Российской Федерации [Электронный ресурс]: режим доступа – http://profstandart.rosmintrud.ru (дата обращения: 1.7.2016).

[5] Государственный стандарт по направлению подготовки 09.03.04 Программная инженерия (программа бакалавриата) [Электронный ресурс]: режим доступа – http://fgosvo.ru/news/2/1086 (дата обращения: 1.7.2016).

он не отражает многообразие реальности, условий и методологий разработки, прежде всего гибких (agile).

Профессиональные стандарты вне и на границах программной инженерии. Вне или на границе программной инженерии находятся профессии из области ИТ:

- связанные с распространением, использованием, обслуживанием программных продуктов и касаются исключительно его потребительских свойств: «Менеджер по ИТ», «Менеджер продуктов в области ИТ», «Специалист по технической поддержке»;
- связанные с поддержкой инфраструктуры и информационным наполнением, такие как «Администратор БД», «Сетевой администратор», «Системный администратор», «Специалист по информационным ресурсам»;
- вспомогательные и второстепенные профессии, связанные с жизненным циклом ПО – «Специалист по тестированию», «Технический писатель», «Специалист по дизайну графического и пользовательского интерфейса». Эта группа обычно перекрывается отдельными дисциплинами направления подготовки бакалавров.

Профессиональный стандарт «Специалист по информационным системам». Содержит 176 страниц текста (для сравнения «Программист» - 18). Содержит очень большой список трудовых функций, в которых требования к знаниям и навыкам повторяются (например, знание «сетевых протоколов» указано в 46 трудовых функциях). Определяет специалиста широкого профиля от менеджера проекта до системного архитектора с довольно поверхностным списком требований.

Профессиональный стандарт «Системный программист». Требования к «штучным» специалистам по разработке таких компонент системного ПО как драйверы, компоненты операционных систем, систем управления базами данных и т.д. В стандарте встречаются такие перлы как «Разработка блок-схемы операционной системы» (стр.12 документа). По содержанию не совместим с образовательными стандартами.

Профессиональный стандарт «Программист» выдержан в русле SWEBOK и привязан к областям знаний (и технологическим процессам) «Проектирование (design)» и «Конструирование (construction)».

Профессиональный стандарт «Системный аналитик» выдержан в русле SWEBOK и привязан к технологическим процессам «Управление требованиями» и «Управление конфигурациями и сопровождение».

Профессиональный стандарт «Архитектор ПО» выдержан в русле SWEBOK и привязан к технологическому процессу «Проектирование (design)».

Профессиональный стандарт «Руководитель разработки» выдержан в русле SWEBOK и привязан к технологическому процессу «Управление программным проектом» с широким перечнем знаний в областях проектирования, конструирования и сопровождения ПО.

Образовательный стандарт бакалавра 09.03.04. Компетентностная модель стандарта выдержана в русле SWEBOK и охватывает большинство описанных там областей знаний (технологических процессов), поэтому в зависимости от наполнения может быть ориентирована на широкий спектр вспомогательных профессий жизненного цикла ПО, отчасти на системного аналитика и архитектора ПО, но в максимальной степени на программиста. Этому способствует и хорошая согласованность требований образовательного и профессионального стандарта «Программист».

Образовательный стандарт магистратуры 09.04.04. Логично было бы предположить, что программа магистратуры по направлению «Программная инженерия»

должна соответствовать профессиям «Архитектор ПО» и «Руководитель разработки». Как минимум, это следует из ориентации на бакалавра-программиста, содержания профессиональных стандартов и общей логике всего направления. Однако образовательный стандарт[6] содержит компетентностную модель в виде перечня областей разработки ПО, в которых магистр способен осуществлять тот или иной вид деятельности. При этом сам перечень фрагментарен и достаточно эклектичен. Справедливости ради следует заметить, что образовательный стандарт магистратуры 09.04.01 «Информатика и вычислительная техника» страдает теми же пороками. Все это слабо согласуется с международными тенденциями образования в этой области [4].

Отсюда следует неутешительный вывод, что в своем текущем виде образовательный стандарт содержит требования к специалистам-исследователям и разработчикам в конкретных предметных областях (аналогично профессиональному стандарту «Системный программист») и слабо привязан к программной инженерии как таковой.

Образовательная программа НГТУ «Программная инженерия» (бакалавр) в значительной степени ориентирована на профессиональный стандарт «Программист»**,** из 108 знаний и умений стандарта в компетентностной модели образовательной программы присутствуют 65 (т.е. 60%). Частично программа перекрывает профессиональный стандарт «Архитектор ПО».

Образовательная программа НГТУ «Программная инженерия» (магистр) учитывает ряд факторов [5]:

- поступающие имеют различный уровень подготовки и компетенций (базовое образование по направлениям ИТ не требуется) – ряд дисциплин обеспечивает «выравнивание» под комплекс знаний программной инженерии;
- сверх требований образовательного стандарта преподается ряд дисциплин в рамках технологических процессов ПИ (системные основы ПИ, методология ПИ, управление программными проектами, средства разработки и поддержки жизненного цикла ПИ и т.п.);
- требования образовательного стандарта обеспечиваются как аудиторным учебным процессом, так и индивидуальной работой в рамках магистерской диссертации.

Почему от кода?

Программная инженерия – дитя нулевых годов нашего века. Производство программного обеспечения превратилось в отдельную отрасль со своими стандартами, разделением труда и производственным циклом. Среди полутора десятков профессий в отрасти ИТ роль профессии программиста можно оценить достаточно противоречиво:

- программист – непосредственный создатель **материальных благ** в этой отрасли. Как бы ни были важны спецификации, модели, документы, без кода, который исполняет задуманное, они бесполезны;
- программист – рабочая лошадка, ремесленник, формальный исполнитель требований, спецификаций и задач, поставленных остальными участниками проекта.

[6] ФГОС ВО по направлениям магистратуры [Электронный ресурс]: режим доступа – http://fgosvo.ru/fgosvo/92/91/5/81 (дата обращения: 1.7.2016).

Очевидно, истина находится где-то посередине, либо вообще плавает по всему диапазону от проекта к проекту. К тому же относительно профессии программиста в самом обществе сложилось немало стереотипов, живучесть которых объясняется сложностью и неочевидностью оценки не только качества и количества труда программиста, но иногда даже его результатов.

Самый последний по времени общественный стереотип – **«тыжпрограммист»**, что программист отвечает за всех и за всё, игнорирует сам факт разделения труда в ИТ. Но с другой стороны, доля истины в нем есть. При разработке кода могут возникать проблемы, корни которых уходят глубоко в «железо», в архитектуру и в функционал, поэтому хороший программист – это все-таки «тыжпрограммист».

Другой интенсивно эксплуатируемый миф о якобы исключительно творческом, креативном характере труда программиста. Корни его уходят в далекие 50-60 годы, когда компьютеры были сродни синхрофазотронам, а работа с ними среднее между священнодействием и трудом физика-теоретика: «Предметом деятельности ученых являются упрощенные модели, в которых они могут абстрагироваться от большинства деталей реального мира, не существенных для их целей… Программист тоже работает с абстракциями, но ему приходится держать в голове гораздо больше абстракций, чем любому ученому…» [6].

Страшно становится от одной мысли: «Где же взять столько одаренных личностей, например, для сопровождения 1С-Бухгалтерии или для разработки сайтов?». Реальность, все-таки проще и прозаичней. Есть разные задачи и разные программисты. Есть традиционная сложившаяся система образования и обучения в лучшем случае программированию. Чем дальше, тем больше требований к продукту труда программиста – коду и к его качеству предъявляется со стороны различных процессов программной инженерии. Рассматривать эти требования абстрактно, вне практики программирования, как минимум, схоластика. Отсюда следует основной тезис: **изложение основ программной инженерии в процессе последовательного усложнения** и **улучшения качества разрабатываемого кода.**

Еще одно замечание. Условно, в сапожном деле можно выделить три категории: те, кто умеет тачать сапоги, и те, которые знают, как это делать и те, которые видели, как это делают другие. В учебном процессе это называется **умения (опыт), знания и представления** соответственно. Программирование в ИТ – то же самое, что электротехника в энергетике. Пожалуй, только «эффективный менеджер» может не знать законов предметной области, в которой он работает, руководствуясь чистым управлением. В разных видах деятельности, связанных с программной инженерией, знание (понимание) процессов, связанных с функционированием программных систем, необходимо в разной степени:

- бизнес-аналитика ПО – не обязательно;
- системная аналитика ПО – желательно, любое требование необходимо оценивать с позиций реализуемости;
- архитектура ПО – безусловно, архитектура – это штучный продукт из доступных решений, каждое из которых нужно уметь оценивать «изнутри»;
- кодирование – по определению;
- тестирование – в зависимости от рода деятельности.

Отдельная позиция по архитекторам

Отдельной темой является архитектура и профессия архитектора. В любой прикладной области архитектура - системное, разностороннее, минимально избыточное описание структуры, поведения и стилей разработки системы (для программных систем известна

модель 4+1 – пять основных «срезов»-описаний системы). Применительно к разработкам в области ИТ можно выделить, как минимум, три вида архитекторов:

- бизнес-архитектор – архитектура бизнес-процессов, экономические, социальные и пр. аспекты разработки программных систем (ПС);
- системный архитектор – создание системы преимущественно из готовых компонент, интеграция, адаптация под бизнес-процессы;
- архитектор ПО – ПС со значительной долей разрабатываемого кода, деятельность, требующая взгляда на программный продукт «изнутри».

Ниже приведена выборка из описания трудовых функций профессионального стандарта «Архитектор программного обеспечения»:

- разработка решений для повторного использования компонентов;
- определение перечня возможных слоев программных компонентов;
- определение перечня возможных протоколов взаимодействия компонентов;
- определение перечня возможных шаблонов (стилей) проектирования для каждого слоя или компонента;
- анализ качества кода: анализ зависимостей, статический анализ кода;
- оценка и выбор модели управления исключениями.

А теперь резонный вопрос: «На каком основании специалист без навыков и опыта работы с кодом может принимать перечисленные решения»? В упомянутом стандарте в качестве требований к образованию указаны направления 230200 «Информационные системы», 230201 (09.03.02) «Информационные системы и технологии», что не совсем логично.

И последнее. Профессия архитектора, как и профессия руководителя, опирается на обобщенный практический опыт, поэтому здесь имеются значительные проблемы в «чистом» образовании. Образно говоря, архитекторами не рождаются, ими становятся.

Восхождение: шаг за шагом

«Многие вещи нам непонятны не потому, что наши понятия слабы; но потому, что сии вещи не входят в круг наших понятий»

Козьма Прутков

Понимание сущности процессов в программной инженерии не может возникнуть на пустом месте и в короткий срок. Для этого необходимо систематическое **восходящее** освоение практики программирования «с нуля» в несколько этапов:

- базовое программирование на Си «на уровне Бейсика»: язык программирования, паттерны анализа и разработки простого программного кода: структурное, «историческое», «грязное» программирование [7];
- продвинутое классическое программирование: структуры данных и алгоритмы, рекурсия, оптимизация по времени и памяти, динамическое программирование, трудоемкость;
- объектно-ориентированное программирование;
- паттерны конструирования, инжиниринг ПО;

- специализированные области разработки ПО, прагматический подход с позиций проектирования (клиент-серверные [8], мобильные приложения, разработка приложений баз данных [9], языковые системы и трансляторы [10]) ;
- основы дисциплин жизненного цикла ПО (1-3 предмета типа «Проектирование ПС», «Конструирование и тестирование ПС», «Управление программными проектами») [11];
- обзорный курс по перспективным тенденциям в средствах разработки ПО (языки – Scala, средства сборки – Maven, Gradle).

Крайне желательно, чтобы каждый этап был подкреплен соответствующей дисциплиной учебного плана (как минимум, 1 семестр на каждую).

Особое внимание следует уделять принципу *преемственности программного кода* в плане его количества и качества, на каждом этапе необходимы конкретные требования по объему и структуре контролируемого в процессе разработки кода.

Проблемы реальные и мнимые

С позиций программной инженерии общие проблемы образования видны особенно ярко. Некоторые из них традиционны для нашего менталитета и образования, другие – приметы нового времени.

Учебные задачи и деловые игры. Отраслевики и инноваторы от учебного процесса постоянно требуют ставить реальные задачи в противовес типовым тематическим заданиям. Безусловно, Двойственное отношение к методическому материалу и учебным задачам. Игры опасны тем, что в них можно реально «заиграться», т.е. упрощенную игровую ситуацию отождествлять с реальностью.

Навыки ремесла. С самого начала обучения дисциплинам цикла программирования многие требования к соблюдению качества кода воспринимаются как придирки со стороны преподавателя. К ним относятся: документирование ключевых моментов разработки, соблюдение стиля кодирования, модульности, отделение функционала от представления, преемственность и повторное использование кода.

Украл или скопипастил. Плагиат (компиляция, копипаст) является большой проблемой не только для объективной оценки выполненной работы, но и для поддержания качественного уровня и дисциплины в учебном процессе. Причем, на всех уровнях, начиная с лабораторных и рефератов и заканчивая курсовыми работами и проектами. В программировании с этим связана еще одна проблема. Для решения задачи можно воспользоваться сторонней библиотекой (закрытым кодом), либо прямо скопировать открытый пример (examlpe), либо адаптировать его под задачу. Методические пособия и публикации на программистских форумах для этого, собственно, и предназначены. Требования преподавателя здесь должны быть сбалансированы и разумны:

- стоит требовать «изобрести велосипед», т.е. своими руками сделать известное решение, если это необходимо для понимания основных принципов организации кода, сущности алгоритмов и структур данных, соблюдения интерфейсов и спецификаций;
- при использовании готовых решений следует оценивать «прибавочную стоимость», т.е. какие изменения внесены в исходный код, насколько студент понимает все нюансы используемого кода, способен ли внести изменения, адаптировать реализацию под новые требования, протестировать и оценить эффективность и затраты ресурсов (время исполнения, память). Обязательно требовать ссылки как на оригинал программного кода, так и на описания его применения.

Ориентация на готовые решения. Еще одна неприятная тенденция общества потребления. Многие студенты убеждены, что любое программное решение можно «нагуглить» в виде подходящей библиотеки. При этом не анализируется, насколько оно соответствует требованиям, перспективно, затратно и т.п.

Уровень абстрагирования. В качественном программном проекте всегда имеется значительная доля абстракций, мета-уровней описания, иерархий слоев и компонент т.п. системных решении. Отсутствие необходимой технологической культуры у студента не позволяет ему не просто начать работу над проектом, но даже просто понять его сущность.

Проблема сложности разработки. Сложность практических и лабораторных заданий связана с ограниченность лимита времени на их решение. Одним из вариантов развития навыков разработки объемного и сложного кода и поддержки его преемственности является лабораторный практикум «нарастающим итогом». На каждом занятии к проекту добавляется новая архитектурная компонента или развивается старая.

Схоластика в образовании тоже вносит свою лепту. Самое очевидное, что структура учебного процесса и наполнение учебных дисциплин консервативны и не отражают текущих реалий. Другой проблемой является отсутствие *рамочных знаний* о предмете: сущность, основные законы и артефакты, границы применимости, другими словами, неразвитость *прагматического подхода.* Зачастую это соседствует с усвоением значительного объема фактического материала. Например, из курса «Математическая логика и теория алгоритмов» студенты знают о машине Тьюринга и даже проектировали ее на практических занятиях, но не знают, что она представляет собой, с одной стороны, исторический артефакт программной системы, а с другой стороны, используется для доказательства алгоритмической разрешимости. В методологическом плане этот формализм лежит в основе правильного понимания проблем автоматизации программирования, тестирования, отладки, т.е. весьма прагматичен.

В отсутствии четко сформулированных рамочных знаний обучение скатывается в одну из крайностей: *доказательство ради доказательства* или *вульгаризация*.

Особенностью программной инженерии является то, что в ней отсутствуют фундаментальные рамочные законы, а во многих случаях достаточны *приближенные и асимптотические оценки* (например, О-нотация в оценке трудоемкости алгоритмов). В связи с этим важна практика оценочного применения математики и упрощенной формализации. Например, студенты решают сложные логарифмические уравнения, знакомые по школе, но не понимают сущности логарифмической шкалы или причин логарифмической трудоемкости алгоритма.

Неразвитость индуктивного подхода. Зачастую в подходе к решению задач используется дедуктивный подход и непродуктивное абстрагирование, в то же время не применяются приемы индуктивного обобщения и установления закономерностей при анализе конкретного поведения изучаемого объекта.

Учеба как самоцель. Инфантилизм и инерция самосознания студентов проявляется в мелочах: «урок» - вместо «пара», «учитель» - вместо «преподаватель», отсутствие навыков самоорганизации. Зачастую это подкрепляется и со стороны учебного процесса: в основных выводах к лабораторным работам или в целях выпускных работ можно видеть «мы изучили, целью является изучение. Необходимо акцентировать, что целью является продукт (разработка), а изучение – накладные расходы, выводы по работе – описание достоинств и недостатков результата и т.п.

Фундамент vs практика. Среди программистов имеется значительное количество практикующих самоучек, с другой стороны, профессиональное образование в программировании не гарантирует успешной работы с этой области. На программистских

форумах периодически вспыхивают дискуссии о пользе или бесполезности фундаментального образования, о достаточности практики использования прогрессивных технологий и средств разработки. Причем у противников фундаментального образования особой нелюбовью почему-то пользуются красно-черные деревья. В данном случае неправы обе стороны. Без отслеживания технологий, безусловно, нельзя. В области разработки ПО удачные разовые решения быстро становятся общедоступными инструментами. В то же время, с помощью любого инструмента можно написать сколь угодно плохую программу, либо использовать его максимально неэффективно, если не знать основных законов его работы.

Итог бакалавриата

Дисциплина «Управление программными проектами» в теории воспринимается достаточно абстрактно. Лабораторный практикум по ней лучше всего проводить в форме *коллективного программного проекта.* Вся группа выполняет единый проект на основе коллективного владения кодом в системе контроля версий. Основное требование: разработка должна разбиваться на относительно независимые подзадачи и легко интегрироваться. Преподаватель выступает в роли менеджера проекта и архитектора, ведется метрика проекта и задач, работа в бригаде ведется по принципам «кодирование/инспекция и документ». Этапы проекта:

- обсуждение бизнес-модели проекта, функционала, программной архитектуры на неформальном уровне (по предварительно составленному *видению проекта*);
- разработка и совместное обсуждение с преподавателем *программного каркаса проекта* (абстракции, интерфейсы, заготовки);
- раздача каркаса под конкретные задачи бригадам студентов и начало его заполнения под контролем преподавателя;
- самостоятельное задач (проектирование, кодинг, отладка, тестирование), интеграция под контролем преподавателя (3-4 занятия).

Кроме всего прочего, такая форма позволяет получить реальные данные для оценки сроков и трудоемкости проекта, по производительности исполнителей, рискам проекта и т.п.

Программная инженерия «для всех»

Естественно, что обучать профессиональной работе с кодом и программной инженерии всех, кто связан с ИТ, не имеет смысла. К тому же в обзорный курс технологий программирования невозможно втиснуть «ремесло» и он остается ознакомительно-обзорным («ты знаешь, мы сегодня на занятии создавали списки»). А вот программная инженерия может дать много полезного сторонним специальностям, в том числе:

- практика ведения проектной деятельности, организации и самоорганизации проектных работ;
- системный и архитектурный подход в проектировании;
- нормативы проектной деятельности, моделирование и документирование предметной области, структуры, поведения (UML).

Заключение

Представленный в статье материал основан на опыте преподавания направления «Программная инженерия» на кафедре вычислительной техники НГТУ. Одной из его оценок

является включение в проект «Лучшие программы инновационной России»[7] и 14-е место НГТУ в Российском рейтинге университетов в номинации «Компьютерные науки»[8].

ЛИТЕРАТУРА

1. Летбридж Т., Лебланк-мл. Р., Собел Э.К., Хилбурн Т., Диас-Херрера Д.. SE2004: рекомендации по обучению специальности «Программная инженерия» // Открытые системы. СУБД. 2006, №10. С. 67-73.
2. ван Влиет Х. О преподавании программной инженерии // Открытые системы. СУБД. 2006, №6. С. 61-66.
3. Орлик С.В. Программная инженерия и жизненный цикл программных проектов с позиций SWEBOK. Часть 1 // Программная инженерия. 2011, №2. С. 6-9.
4. Сухомлин В.А. Анализ международных стандартов магистерского образования в области информационных технологий / Вестн. С.-Петербург. ун-та. Сер. 10. Прикл. матем. информ. проц. упр., 2013, выпуск 1, С. 95–105.
5. Губарев В.В., Михеенко О.И., Перфильев Ю.С., Романов Е.Л. Обучение в магистратуре (проблемы и предложения) // Современное образование: перспективы развития многопрофильного технического университета: материалы междунар. науч.-метод. конф., Томск, 28–29 янв. 2010 г. – Томск : Изд-во ТУСУР, 2010. – С. 49–50.
6. Архипенков С. Лекции по управлению программными проектами [Электронный ресурс]: режим доступа – http://www.arkhipenkov.ru/ (дата обращения: 01.07.2016).
7. Романов Е.Л. Практикум по программированию на C++: Уч. пособие. СПб: БХВ-Петербург; Новосибирск: Изд-во НГТУ, 2004. - 432 с.
8. Романов Е.Л. Архитектура и прикладные протоколы клиент-серверных приложений. Электронный учебно-методический комплекс. № ОФЭРНиО: 21514 [Электронный ресурс]: режим доступа – http://dispace.edu.nstu.ru/didesk/course/show/5379 (дата обращения: 01.07.2016).
9. Разработка приложений на C# с использованием СУБД PostgreSQL: учебное пособие / И.А. Васюткина, Г.В. Трошина, М.И. Бычков, С.А. Менжулин. – Новосибирск: Изд-во НГТУ, 2015. – 143 с. ISBN 978-5-7782-2699-9.
10. Романов Е.Л. Теория языков программирования и методы трансляции. Электронное учебное пособие для студентов 3 курса дневного и заочного отделений направления 230100. Новосибирский государственный технический университет, № гос. регистрации 0321100595., 2011, 150 с.
11. Романов Е.Л. Программная инженерия. Электронный учебно-методический комплекс [Электронный ресурс]: режим доступа – http://dispace.edu.nstu.ru/didesk/course/show/5164 (дата обращения: 01.07.2016).

[7] Официальный сайт проекта «Лучшие инновационные программы России» [Электронный ресурс]: режим доступа – http://www.best-edu.ru/spr/searchBestProgramm?years%5B%5D=2014&region%5B%5D=54&vuzs%5B%5D=556&spec%5B%5D=Программная+инженерия&find= (дата обращения: 1.7.2016).

[8] Рейтинг факультетов: предметный рейтинг университетов России // Эксперт – 2016, – №23 (990), С. 57-67.

Romanov Evgeniy Leonidovich
Novosibirsk state technical university, Russia, Novosibirsk
E-mail: romanov@corp.nstu.ru

Troshina Galina Vasil'evna
Novosibirsk state technical university, Russia, Novosibirsk
E-mail: troshina@corp.nstu.ru

Yakimenko Alexander Aleksandrovich
Novosibirsk state technical university, Russia, Novosibirsk
E-mail: al--le@yandex.ru

Program engineering teaching. View from the code

Abstract. The higher education aspects in the "Program Engineering" preparation direction are considered in article. Historical continuity with the "Informatics and Computer Engineering" direction is noted. There historical continuity with the direction of "Computer Science and Engineering". The detailed analysis and the critical evaluation of the professional standards in the field of the information technologies and the educational standards of the bachelor's training and the master's training from the position of the program engineering body of knowledge is carried out. It is noted that the bachelor educational standard in general is coordinated with the "Programmer" profession standard requirements and is sustained in the knowledge course. The master's educational standard shortcomings complicating the professions preparation of this row - "The software architect" and "Head of the development" are analyzed. The "from a code" training methodology based on the programming practice consecutive complication is proved, her stages are listed. The balance of the creative and the routine beginnings in the profession and the developed stereotypes are discussed. It is shown that requirements for the software architect can't be executed without the profound knowledge and the practice of a koding and a software design and are in the "Program Engineering" direction course. In relation to this direction the common "sore points" of the students psychology and the educational process organization are discussed: the educational tasks vs the business games, the copy-paste, the orientation to ready decisions, the abstraction level, the study as end in itself, the development complexity problem, the inductive approach backwardness, the education scholasticism, the base vs the practice. Useful parties of discipline "Program Engineering" for the related directions are noted.

Keywords: software engineering; professional standards; educational standards; programmer; coding; programming and software design practice; educational process organization; bachelor's training; the master's training; "from a code" training methodology