Рекомендации по преподаванию программной инженерии и информатики в университетах

Software Engineering 2004: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering

Computing Curricula 2001: Computer Science

перевод с английского

Москва, 2007

УДК [004.9+004.438](072)
ББК 32.81я81+32.973.26-018я81
Р36

Р36 Рекомендации по преподаванию программной инженерии и информатики в университетах = Software Engineering 2004: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering; Computing Curricula 2001: Computer Science: пер. с англ. — М.: ИНТУИТ.РУ «Интернет-Университет Информационных Технологий», 2007. — 462 с. : ил. – Данные тит. л. частично парал. англ.

ISBN 978-5-9556-0105-9.

Книга представляет собой заключительный отчет специальной объединенной комиссии ACM и IEEE Computer Science, содержащий рекомендации по преподаванию программной инженерии и информатики и типовым учебным планам этих дисциплин.

Книга будет полезна преподавателям и студентам в области программной инженерии и информатики.

Перевод SE2004: Н.И. Бойко, М.Е. Зверинцева, С.А. Алпаев, Д.А. Маленко, И.В. Мозговая
Редакторы перевода SE2004: В.Л. Павлов, А.А. Терехов, А.Н. Терехов

Перевод CC2001: М.Е. Зверинцева, Т.В. Зверинцева, Н.Ю. Курочка, А.А. Симановский, Д.А. Шапоренков
Редакторы перевода CC2001: В.Л. Павлов, А.А. Терехов

Рекомендовано к изданию кафедрой системного программирования Санкт-Петербургского Государственного Университета

Перевод и издание подготовлены при финансовой поддержке Ассоциации Предприятий Компьютерных и Информационных Технологий (АПКИТ).

АССОЦИАЦИЯ ПРЕДПРИЯТИЙ КОМПЬЮТЕРНЫХ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Первое издание: Санкт-Петербургский Государственный Университет, 2002

ISBN 978-5-9556-0105-9

Содержание

Рекомендации по преподаванию программной инженерии в университетах

Software Engineering 2004: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering

От редакторов перевода

Вашему вниманию предлагается перевод на русский язык документа «Software Engineering 2004: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering», в котором собран всемирный опыт преподавания программной инженерии в университетах и колледжах.

История проекта Computing Curricula, в рамках которого был выпущен данный документ, ведет свой отсчет с 1968 года, когда была опубликована первая версия рекомендаций по преподаванию информатики в университетах. С тех пор эти рекомендации обновлялись примерно раз в десять лет совместным комитетом по образованию под эгидой профессиональных ассоциаций ACM (Association for Computing Machinery) и IEEE Computer Society.

В конце 1990-х годов стало ясно, что область знаний, связанная с информационными технологиями, очень сильно разрослась и ее трудно, если вообще возможно, полностью осветить в рамках одного университетского курса. В связи с этим было принято решение о ее разделении на четыре основные дисциплины – информатика (computer science), программная инженерия (software engineering), проектирование аппаратных платформ (hardware engineering) и информационные системы (information systems).

Первый том в серии Computing Curricula 2001, посвященный информатике, был выпущен в конце 2001 года. В качестве официальных рекомендаций по преподаванию информационных систем был утвержден документ «Information Systems 2002», разработанный в результате совместного проекта ACM, AIS (Association for Information Systems) и AITP (Association of Information Technology Professionals). Рекомендации по преподаванию программной инженерии были выпущены в августе 2004 года. Наконец, документ с рекомендациями по преподаванию проектирования аппаратных платформ был утвержден в декабре 2004 года.

В сентябре 2005 года был выпущен обзорный том для всего проекта Computing Curricula. В нем была впервые сформулирована потребность выделения еще одной самостоятельной дисциплины под названием «информационные технологии» (information technology). В ближайшие годы ожидается начало следующей итерации обновления стандартов серии Computing Curricula, возможно, она приведет к дальнейшему расширению списка дисциплин.

Редакторы данного перевода познакомились с Computing Curricula еще в 1990-х годах. В 1996 году рекомендации Computing Curricula 1991 были использованы проф. Андреем Николаевичем Тереховым в качестве методологической базы для описания направлений преподавания и исследований на вновь создаваемой кафедре системного программирования Санкт-Петербургского государственного университета.

В 2001-02 годах Владимир Павлов и Андрей Терехов-младший реализовали проект по переводу, изданию и рассылке по университетам стран СНГ русской версии документа Computing Curricula 2001: Computer Science. Так как русский пе-

ревод этого документа был выпущен тиражом всего 600 экземпляров и не поступал в продажу, книга мгновенно стала библиографической редкостью. Поэтому мы приняли решение опубликовать ее полный текст с незначительными уточнениями и исправлениями в данном томе, вместе с переводом Software Engineering 2004. Это решение подкреплено также наличием тесных связей между программной инженерией и информатикой. Часто говорят, что программная инженерия относится к информатике так же, атомная энергетика к физике. Несмотря на некоторую рискованность такой метафоры, в ней заложен глубокий смысл. В современном мире невозможно стать высококвалифицированным программным инженером без знания информатики. Однако столь же глубоким заблуждением было бы полагать, что для подготовки грамотных программных инженеров достаточно изучения одной информатики. Это взаимосвязанные, но все-таки существенно различающиеся области знаний.

Скажем несколько слов о процессе перевода Software Engineering 2004 на русский язык. Первый черновик перевода был готов еще осенью 2005 года, однако по разным причинам Владимир Павлов и Андрей Терехов-младший не смогли в тот момент уделить этому проекту достаточно времени, и работа над переводом была заморожена на полгода. В этот момент к участию в проекте подключился профессор Андрей Николаевич Терехов, внесший в итоге неоценимый вклад в редактирование перевода.

Как это ни странно, одной из основных трудностей проекта стал перевод слова «computing», обозначающего обобщенную область знаний, в которую входят информатика, программная инженерия, проектирование аппаратных платформ и прочие дисциплины, так или иначе связанные с информационными технологиями. Это слово можно примерно перевести как «вычислительные науки» или «вычислительная техника», но, к сожалению, оба этих термина слишком узки, т.к. computing включает в себя и науку, и технику, и инженерные дисциплины. От варианта «информационные технологии» тоже пришлось отказаться, так как информационные технологии являются лишь одной из самостоятельных дисциплин в рамках более широкой области знаний под названием computing. В связи с этим мы приняли непростое решение об использовании при переводе транслитерации английского слова, т.е. «компьютинг». Этот перевод не идеален, но позволяет избежать двойственного толкования терминов и потому постепенно входит в употребление среди российских ИТ-специалистов. Кроме того, этот перевод используется в широко известном «Англо-русском толковом словаре по вычислительной технике, Интернету и программированию» под редакцией Э. Пройдакова и Л. Теплицкого (4-е издание).

В Software Engineering 2004 приводится множество шаблонов составления учебных программ для разных стран с учетом их традиций преподавания. К сожалению, среди этого множества нет шаблона, учитывающего особенности российского образования. Поэтому в процессе работы над переводом А.А. Терехов и А.Н. Терехов подготовили собственное предложение по совмещению Software

Engineering 2004 с российскими образовательными стандартами, представили его на нескольких конференциях, посвященных ИТ-образованию, и опубликовали его в качестве статьи в журнале «Открытые системы», №8, 2006 (см. http://www.osp.ru/os/2006/08/3282281).

К сожалению, в университетах России и стран СНГ так и не произошло становления программной инженерии как самостоятельной дисциплины. Мы надеемся, что публикация данного перевода послужит катализатором для появления и распространения в России учебных стандартов по программной инженерии.

В заключение, мы хотели бы поблагодарить спонсоров данного проекта, оказавших данному проекту организационную и финансовую поддержку:

- Ассоциация Предприятий Компьютерных и Информационных Технологий (АП КИТ)
- Intel Россия
- Microsoft Россия

Оригинальный английский текст Software Engineering 2004 можно скачать с сайта IEEE Computer Society по адресу
http://www.computer.org/portal/cms_docs_ieeecs/ieeecs/education/cc2001/SE2004Volume.pdf.

В. Л. Павлов (Международный институт эффективных информационных технологий INTSPEI), vladimir@vlpavlov.com
к.ф.-м.н. А.А. Терехов (Microsoft), andrey@terekhov.net
проф., д.ф.-м.н. А.Н. Терехов (Санкт-Петербургский государственный университет, ЛАНИТ-ТЕРКОМ), ant@tercom.ru

Предисловие

Данный документ создан в рамках совместного проекта Образовательного совета ACM (ACM Education board) и Совета по образовательной деятельности компьютерного сообщества IEEE (IEEE Computer Society Educational Activities Board), целью которого является разработка рекомендаций к учебным планам по ряду дисциплин компьютинга (computing): информатика (computer science), проектирование аппаратных платформ (computer engineering), программная инженерия (software engineering) и информационные системы (information systems). В работе над отдельными проектами из этого списка принимали участие и другие профессиональные организации. Ярким примером такого сотрудничества является представленный в данном документе проект SE2004 («Программная инженерия 2004»), в котором принимали участие представители Австралийского компьютерного сообщества (Australian Computer Society), Британского компьютерного сообщества (British Computer Society), а также Японского сообщества по обработке информации (Information Processing Society of Japan).

Процесс разработки

Проект SE2004 осуществлялся под руководством организационного комитета, назначенного организациями-спонсорами. Работа над проектом началась осенью 2001 года с назначения сопредседателей и нескольких участников. В первой половине 2002 года количество участников комитета возросло, в том числе, и за счет представителей других сообществ. Ниже перечислены члены организационного комитета SE2004:

Сопредседатели:

Rich LeBlanc, ACM, Georgia Institute of Technology, U.S.

Ann Sobel, IEEE-CS, Miami University, U.S.

Председатель группы по структуризации преподаваемых знаний

Ann Sobel, Miami University, U.S.

Сопредседатели группы по педагогике

Mordechai Ben-Menachem, Ben-Gurion University, Israel

Timothy C. Lethbridge, University of Ottawa, Canada

Редакторы

Jorge L. Diaz-Herrera, Rochester Institute of Technology, U.S.

Thomas B. Hilburn, Embry-Riddle Aeronautical University, U.S.

Представители организаций:

ACM: Andrew McGettrick, University of Strathclyde, U.K.

ACM SIGSOFT: Joanne M. Atlee, University of Waterloo, Canada

ACM Two-Year College Education: Elizabeth K. Hawthorne, Union County College, U.S.

Australian Computer Society: John Leaney, University of Technology Sydney, Australia
British Computer Society: David Budgen, Keele University, U.K.
Information Processing Society of Japan: Yoshihiro Matsumoto, Musashi Institute of Technology, Japan
IEEE-CS Technical Committee on Software Engineering: J. Barrie Thompson, University of Sunderland, U.K

Благодарности

Документ был создан при поддержке Национального научного фонда США (National Science Foundation), Ассоциации по вычислительной технике (Association of Computing Machinery) и Компьютерного сообщества IEEE (IEEE Computer Society).

Свой вклад в данный проект со времени его начала внесли многие люди, некоторые из которых выступили сразу в нескольких ипостасях. Данный проект вряд ли мог быть осуществлен без самоотверженности и знаний этих добровольцев. Имена специалистов, участвовавших в написании и рецензировании различных версий этого документа, приведены в приложении Б. Мы также хотели бы выразить особую благодарность Susan Mengel из Техасского технического университета, которая была первым сопредседателем оргкомитета и выполнила начальную работу по организации проекта SE2004.

ГЛАВА 1. Введение

1.1. Цель данного документа

Основной целью данного документа является разработка рекомендаций для образовательных учреждений и агентств по аккредитации содержания учебных планов для подготовки бакалавров в области программной инженерии. Эти рекомендации были разработаны группой добровольцев, состоящей из большого числа специалистов из разных стран. При составлении данного документа учитывались результаты, полученные в области обучения программной инженерии за последние 25 лет. В настоящий момент рекомендации к составлению учебных планов по программной инженерии особенно актуальны, так как в ряде стран резко возросло количество вновь создаваемых программ обучения программной инженерии, и возникает потребность в процессе аккредитации этих программ.

Рекомендации, содержащиеся в данном документе, разработаны на основе высокоуровневых требований к знаниям, которыми должен обладать выпускник, обучавшийся в вузе по специальности «Программная инженерия». Из этих требований, представленных в главе 3, логически вытекают основные содержательные результаты данного документа:

- Преподаваемый материал по программной инженерии (Software Engineering Education Knowledge, также будет использоваться аббревиатура SEEK) – что должен знать каждый выпускник вуза, специализирующийся по программной инженерии.
- Учебный план – описание различных подходов и методик, с помощью которых эти знания и фундаментальные для программной инженерии навыки могут преподаваться в различных контекстах.

1.2. Как данный том соотносится с Computing Curriculum

Для решения данной проблемы рабочая группа продолжила работу над томом, содержащим рекомендации по преподаванию информатики, который был опубликован в 2001 году под названием Computing Curricula 2001: Computer Science (далее – CCCS; в русском переводе – «Рекомендации по преподаванию информатики в университетах») [ACM 2001]. Кроме того, она рекомендовала организациям-спонсорам расширить проект путем создания отдельных томов с рекомендациями по родственным дисциплинам, перечисленным выше, а также по любым другим дисциплинам, которые будут признаны необходимыми образовательным сообществом. Данный документ представляет собой результат совместного проекта ACM и IEEE-CS по созданию рекомендаций по преподаванию программной инженерии (Software Engineering 2004 или, сокращенно, SE2004) и является первой попыткой зафиксировать содержание и методику преподавания в данной области.

В конце 2002 года был одобрен и опубликован «Типовой учебный план и рекомендации по преподаванию специальности «Информационные системы» в университетах» – документ, разработанный совместной рабочей группой ACM, Ассоциации информационных систем (Association for Information Systems, AIS), а также Ассоциации профессионалов в области информационных технологий (Association of Information Technology Professionals, AITP). Кроме того, разрабатываются рекомендации по преподаванию специальностей «Проектирование аппаратных платформ» и «информационные технологии».

1.3. Процесс разработки SE2004

Разработка данного документа осуществлялась в три основных этапа с привлечением большого количества добровольцев, а также всех членов организационного комитета. Первый этап включал в себя разработку набора ожидаемых результатов обучения и объема знаний, необходимых каждому выпускнику, обучавшемуся по специальности «Программная инженерия». В рамках второго этапа были определены и классифицированы знания, которые должны быть включены в преподаваемый в университетах материал по программной инженерии (SEEK). На третьем этапе был разработан набор рекомендаций по созданию учебных планов, описывающий, каким образом учебный план по программной инженерии, включающий в себя SEEK, может быть структурирован в различных контекстах.

1.3.1. Рабочая группа по определению преподаваемого материала

Весной 2002 года была создана рабочая группа, которой было поручено разработать первую версию преподаваемого материала по программной инженерии (Software Engineering Education Knowledge, SEEK). Группе была передана начальная структуризация дисциплины программной инженерии, снабженная кратким описанием каждой области, и поручено определить модули и темы для каждой области знаний с использованием шаблонов, разработанных организационным комитетом. В качестве дополнительной информации разработчики SEEK использовали результаты открытого семинара, проведенного в рамках CSEE&T 2002 (Конференция по обучению и тренингу в области программной инженерии) [Thompson 2002], а также результаты обсуждений обязательного содержания учебных планов в рамках саммита по преподаванию программной инженерии (Summit on Software Engineering Education), который проводился во время ICSE (Международной конференции по программной инженерии) [Thompson 2004].

Первичные результаты рабочей группы были включены в предварительную версию документа SEEK. Дальнейшая работа над этим документом проводилась в рамках специального семинара по SEEK, организованного Национальным научным фондом США в июне 2002. В данном семинаре участвовали члены группы по структуризации знаний, члены организационного комитета, ведущие специали-

сты в области обучения программной инженерии и приглашенные представители рабочей группы по педагогике. Результаты работы семинара были впоследствии доработаны организационным комитетом.

Полученное описание SEEK было передано на выборочное рецензирование группе всемирно известных экспертов в области программной инженерии. Полученные оценки и комментарии были использованы Организационным комитетом для подготовки первой официальной черновой версии SEEK, опубликованной для общего рецензирования в августе 2002 года.

К моменту окончания первой сессии рецензирования в начале октября 2002 года в организационный комитет поступило около 40 рецензий. Организационный комитет дал письменный ответ на каждую рецензию, включая планируемые действия и их обоснование. После публикации второй версии SEEK в декабре 2002 была проведена следующая сессия рецензирования, окончившаяся в марте 2003 года. Рабочая группа по вопросам обучения и тренинга в области программной инженерии (Working Group on Software Engineering Education and Training, WGSEET) оказала значительную помощь в процессе приведения содержания второй версии SEEK в соответствие с рекомендациями рабочей группы по педагогике. Вклад WGSEET наряду с результатами второй сессии рецензирования привел к появлению окончательной версии SEEK.

1.3.2. Рабочая группа по педагогике

В октябре 2002 года рабочая группа по педагогике начала разработку рекомендаций по учебным планам в области программной инженерии, взяв за основу SEEK. Был сформирован план и определен процесс работы группы. Члены группы начали работу по определению педагогических рекомендаций, типовых учебных планов, адаптации под схемы обучения в различных странах, а также вопросы среды внедрения. Полученные данные были уточнены организационным комитетом в феврале 2003 года. Рецензирование чернового варианта отчета рабочей группы по педагогике было проведено на заседании WGSEET и на семинаре, прошедшем в марте 2003 года в рамках конференции по обучению и тренингу в области программной инженерии.

Предварительная версия отчета по педагогике содержала следующие разделы:

- Принципы составления и использования учебных планов по программной инженерии.
- Предлагаемый учебный план, включающий в себя различные модели учебных планов и примеры курсов, включая информацию о том, какие разделы SEEK покрываются данным курсом.
- Вопросы адаптации планов к схемам обучения в различных странах.
- Классификацию навыков, которые студенты должны получить, и проблем, которые они должны уметь решать, в дополнение к знаниям, определенным в SEEK.

- Адаптация планов к альтернативной среде обучения, например, колледжам с двухгодичной программой обучения.

Типовые учебные планы были разработаны с использованием SEEK, тома Computing Curricula, посвященного информатике (CCCS), а также результатов обзора существующих учебных программ по подготовке бакалавров. Для достижения этой цели были определены и изучены 32 образовательные программы Северной Америки, Европы и Австралии.

Основной техникой при создании типового учебного плана было определение тем SEEK, которые могли бы быть покрыты уже существующими курсами из CCCS.

В основе метода разработки моделей лежало определение того, какие темы SEEK будут раскрыты при использовании существующих курсов CCCS. Остальной материал SEEK был распределен по курсам по программной инженерии, используя существующие учебные программы в качестве примеров.

1.3.3. Окончательная разработка документа

Весной и летом 2003 года были написаны некоторые дополнительные материалы (введение, рекомендации и желаемые результаты, общие сведения о программной инженерии и т.п.), которые вместе с SEEK и учебными планами составили полный текст документа SE2004. Первое рецензирование SE2004 проводилось на Втором саммите по преподаванию программной инженерии (Second Summit on Software Engineering Education), проходившем в рамках конференции ICSE 2003 [Thompson 2003]. Результаты работы саммита и неформальные отзывы были использованы организационным комитетом для создания первой официальной версии SE2004, проходившей рецензирование с июля по сентябрь 2003 года. Свои замечания и рекомендации представили также Совет по ACM и Совет по образовательной деятельности IEEE-CS. В результате анализа организационным комитетом полученных комментариев был сформирован окончательный проект документа SE2004.

1.4. Структура документа

В главе 2 обсуждается содержание программной инженерии как дисциплины, приводятся некоторые фрагменты истории преподавания программной инженерии и объясняется, как эта история повлияла на рекомендации, приведенные в данном документе. Глава 3 представляет основополагающие принципы разработки данного документа. Эти принципы были адаптированы из принципов, сформулированных рабочей группой CC2001 в начале работы над документом, получившим впоследствии название CCCS. Глава 3 содержит также описание знаний, которыми должен обладать каждый выпускник по специальности «Программная инженерия». Глава 4 представляет преподаваемый материал по про-

граммной инженерии (SEEK), являющийся основой для рекомендаций по преподаванию и созданию учебных программ, которые представлены в главах 5 и 6 соответственно. В главе 7 обсуждаются вопросы адаптации рекомендаций по составлению учебных планов к альтернативным средам обучения. Наконец, в главе 8 обсуждаются различные трудности внедрения учебных планов и предложены подходы к оценке учебных планов.

ГЛАВА 2. Дисциплина программной инженерии

В данной главе обсуждается природа программной инженерии, история этой дисциплины и предпосылки к созданию рекомендаций по составлению учебных планов по программной инженерии. Цель данной главы – дать контекст и логическое обоснование для материалов учебных планов, изложенных в последующих главах.

2.1. Дисциплина программной инженерии

Со времени возникновения вычислительной техники в 1940-х годах применение и использование компьютеров развивались ошеломляющими темпами. Программное обеспечение играет важную роль практически во всех аспектах повседневной жизни: государственном управлении, банковском деле и финансах, образовании, транспорте, индустрии развлечений, медицине, сельском хозяйстве и юриспруденции. Количество, размеры и области применения компьютерных программ резко увеличились. В результате сотни миллиардов долларов затрачиваются на разработку программного обеспечения, и от эффективности этих программ зависят заработки и даже жизни большинства людей. Программные продукты помогли нам стать эффективнее и продуктивнее. Они помогают в решении задач и предоставляют среду для работы и развлечений, во многих случаях более защищенную, более гибкую и менее ограничивающую. Однако, несмотря на все эти успехи, достижение адекватной стоимости, сроков разработки и качества программных продуктов является серьезной проблемой. Существует множество причин возникновения проблем, включая следующие:

- Программные продукты относятся к самым сложным системам, которые создаются человеком, и программное обеспечение по самой своей природе обладает рядом существенных и неотъемлемых свойств (таких как сложность, незримость и изменяемость), которые затрудняют работу [Brooks 95].
- Методы и процессы программирования, которые эффективно работают для одного человека или для небольшой команды при разработке программ умеренных размеров, плохо масштабируются для разработки крупных и сложных систем (т.е. систем, состоящих из миллионов строк кода и требующих нескольких лет работы сотен разработчиков программного обеспечения).
- Скорость изменения компьютерных и программных технологий создает потребность в новых и эволюционирующих программных продуктах. Пользовательские ожидания и конкурентная борьба, возникающие в таких условиях, существенно затрудняют возможность выпускать качественное программное обеспечение в приемлемые сроки.

Прошло более 35 лет с момента первой организованной формальной дискуссии о программной инженерии как научной дисциплине на Конференции НАТО по программной инженерии [Naur 1969], состоявшейся в 1968 году. Термин «программная инженерия» сейчас широко используется в индустрии, государственных и учебных учреждениях – сотни тысяч специалистов именуют себя «программными инженерами» (software engineers), термин «программная инженерия» фигурирует в названиях множества публикаций, групп, организаций и профессиональных конференций, существует множество учебных курсов и программ обучения программной инженерии. Однако по-прежнему существуют разногласия и различные мнения по значению данного термина. Приведенные ниже определения дают несколько различных представлений о значении и природе программной инженерии. Тем не менее им всем присуща одна общая черта: все они сходятся в том, что программная инженерия – нечто большее, чем просто написание программного кода (coding) и включает в себя аспекты качества, управления и экономики, а также знание и применение на практике этих принципов и дисциплин.

Определения программной инженерии

В течение многих лет давались различные определения дисциплины программной инженерии. В целях данного документа мы отметим следующие определения:

- «Установление и использование правильных инженерных принципов (методов) для экономичного получения надежного и работающего на реальных машинах программного обеспечения» [Bauer 1972].
- «Программная инженерия является такой формой инженерии, которая применяет принципы информатики (computer science) и математики для получения рентабельных решений в области программного обеспечения» [CMU/SEI-90-TR-003].
- «Применение систематического, дисциплинированного, поддающегося количественному определению подхода к разработке, эксплуатации и сопровождению программного обеспечения» [IEEE 1990].

Каждое из этих определений содержит отдельные аспекты, повлиявшие на общее понимание программной инженерии, представленное в данном документе. Одно из наиболее важных наблюдений состоит в том, что программная инженерия основывается на информатике и математике. Однако, в духе инженерных традиций, она выходит за рамки этого технического базиса и использует результаты более широкого диапазона дисциплин.

Данные определения явно формулируют, что программная инженерия посвящена систематическим, управляемым и эффективным методам создания высококачественного программного обеспечения. Поэтому особое внимание уделяется анализу и оценке, спецификации, проектированию и эволюции программ-

ного обеспечения. Кроме того, в рамки данной дисциплины попадают вопросы, связанные с управлением и качеством, новизной и творчеством, стандартами, индивидуальными навыками и командной работой, а также профессиональной деятельностью, которые играют жизненно важную роль в программной инженерии.

2.2. Программная инженерия как одна из дисциплин компьютинга

Часто встречающимся заблуждением о программной инженерии является представление о том, что она преимущественно связана с деятельностью, ориентированной на процессы (например, управление требованиями, проектирование, обеспечение качества, совершенствование процессов и управление проектом). С этой точки зрения для достижения компетентности в области программной инженерии достаточно иметь инженерную подготовку, представлять в общих чертах процесс разработки программного обеспечения и обладать минимальными познаниями в компьютинге, включая навыки использования одного или нескольких языков программирования. Однако на практике таких знаний совершенно недостаточно, а заблуждение, приводящее к подобной точке зрения, основано на неполном представлении о природе и проблемах программной инженерии.

В ходе развития вычислительной техники исторически сложилось, что специалисты по компьютерным наукам писали программное обеспечение, а электронщики производили аппаратное обеспечение, на котором работали программы. По мере увеличения размеров, сложности и критичности программного обеспечения возрастала и потребность в соответствии программного обеспечения исходным требованиям. К началу 1970-х годов стало очевидно, что для грамотной разработки программного обеспечения недостаточно простого применения основных принципов информатики – требуются аналитические и описательные средства, разработанные специалистами по информатике, и тщательность, с которой инженерные дисциплины добиваются надежности и достоверности создаваемых ими артефактов.

Таким образом, программная инженерия качественно отличается от других инженерных дисциплин нематериальностью программного обеспечения и дискретной природой его функционирования. Программная инженерия стремится интегрировать принципы математики и информатики с инженерными подходами, разработанными для производства осязаемых материальных артефактов. Основываясь на математике и компьютинге, программная инженерия занимается разработкой систематических моделей и надежных методов производства высококачественного программного обеспечения, и данный подход распространяется на все уровни – от теории и принципов до реальной практики создания программного обеспечения, которая лучше всего заметна сторонним наблюдателям. Хотя и не предполагается, что каждый программный инженер обладает глубокими знаниями во всех аспектах компьютинга, общее понимание их применимости

и квалификация в каких-то определенных областях являются абсолютно необходимыми. Определение преподаваемого материала по программной инженерии (SEEK), приведенное в главе 4, отражает зависимость программной инженерии от информатики (в частности, самой крупной компонентой SEEK являются «Основы компьютинга»).

2.3. Программная инженерия как инженерная дисциплина

На исследования и практику в области программной инженерии влияют как ее корни в информатике, так и ее установление в качестве самостоятельной инженерной дисциплины. Большая часть исследований в области программной инженерии проводится на факультетах или кафедрах информатики или вычислительной техники. Схожим образом, учебные программы по программной инженерии разрабатываются как на факультетах информатики, так и в инженерных колледжах. Таким образом, дисциплина программной инженерии может рассматриваться как инженерная область, имеющая более тесные связи со своей базовой дисциплиной (информатикой), чем другие инженерные области. При составлении данного тома особое внимание уделялось включению инженерных приемов в разработку программного обеспечения, чтобы обозначить различие между данным учебным планом и учебным планом по информатике. В качестве подготовки к более подробному развитию этих идей, в данном разделе рассматривается инженерная методология и ее применение к разработке программного обеспечения.

Мы также должны отметить, что одновременно со значительным сходством между программной инженерией и традиционной инженерией (как описано в разделе 2.3.1), существуют и некоторые отличия (не обязательно в ущерб программной инженерии):

- Основанием программной инженерии является информатика, а не естественные науки.
- Основной упор делается на дискретной, а не на непрерывной математике.
- Концентрация на абстрактных/логических объектах вместо конкретных/физических артефактов.
- Отсутствие «производственной» фазы в традиционном промышленном смысле.
- «Сопровождение» программного обеспечения в основном связано с продолжающейся разработкой или эволюцией, а не с традиционным физическим износом.

2.3.1. Характеристики инженерии

Существует набор характеристик, являющихся не только общими для всех инженерных дисциплин, но и настолько существенных и критических, что они

могут использоваться для описания основ инженерии как таковой. Именно такие характеристики должны рассматриваться как желательные для всех программных инженеров (software engineers). Этот набор характеристик оказал существенное влияние как на развитие программной инженерии, так и на содержание данного документа.

[1] Инженеры в своей деятельности принимают ряд решений, тщательно оценивая альтернативы и выбирая в каждой точке принятия решения подход, оптимально соответствующий решаемой задаче с учетом существующего контекста. Выбор подхода осуществляется в процессе анализа альтернатив, во время которого тщательно сопоставляются возможные затраты и ожидаемая прибыль.

[2] Инженеры, по возможности, работают с использованием измеримых количественных характеристик; они совершенствуют и уточняют существующие методы измерений и при необходимости выдают приближенные решения на основе опыта и эмпирических данных.

[3] Инженеры придают особое значение использованию дисциплинированного процесса при осуществлении проекта и понимают важность вопросов эффективной организации командной работы.

[4] Инженеры могут отвечать за выполнение самого широкого спектра задач, начиная с исследований, разработки, проектирования, производства, тестирования, внедрения, эксплуатации и управления, и заканчивая продажами, консультированием и обучением.

[5] Инженеры в процессе выполнения своих обязанностей широко используют инструментальные средства. Поэтому выбор и использование походящих средств является крайне важным вопросом.

[6] Объединяясь в профессиональные сообщества, инженеры способствуют развитию своей отрасли путем разработки и внедрения рекомендаций, аттестационных принципов, стандартов, распространению хорошо зарекомендовавших себя подходов (best practices).

[7] Инженеры повторно используют (reuse) результаты проектирования и проектные артефакты.

Следует подчеркнуть, что хотя термины «инженер» и «инженерия» часто используются в последующих разделах, данный документ, прежде всего, посвящен вопросам проектирования, разработки и реализации университетских учебных планов по программной инженерии. Также необходимо отметить, что большая часть информации в этом документе основывается на работах значительного количества специалистов и групп, которые способствовали становлению информатики и информационных технологий и благодаря кропотливому труду которых многие десятки тысяч выпускников университетов влились в ряды профессиональных разработчиков программного обеспечения.

2.3.2. Инженерное проектирование

Проектирование является важной составляющей любой инженерной деятельности и играет критически важную роль при разработке программного обеспечения. Деятельность в рамках инженерного проектирования связана с созданием новых артефактов путем нахождения технических решений для специфических практических задач с учетом экономических, юридических и социальных соображений. Инженерное проектирование, по существу, предоставляет необходимые условия для «физической» реализации решения, что достигается путем следования систематическому процессу, который наилучшим образом удовлетворяет набору поставленных требований в рамках потенциально противоречивых ограничений.

Программная инженерия отличается от традиционной инженерии особой природой программного обеспечения, в связи с чем основной упор делается на абстракцию, моделирование, организацию и представление информации, а также на управление изменениями. Программная инженерия также включает в себя деятельность по реализации проекта и контролю качества, которая в традиционном инженерном цикле обычно относится к фазам проектирования производственного процесса и производства. Кроме того, непрерывная эволюция (т.е. «сопровождение») также является критически важной для программного обеспечения.

Даже при столь обширном охвате тем центральной задачей программной инженерии является инженерное проектирование – разновидность деятельности по принятию решений. Важным аспектом данной задачи является необходимость применения соответствующих процессов на многочисленных уровнях абстракции. Возрастающая популярность подходов, основанных на повторном использовании компонент, вселяет надежду на появление новых, улучшенных способов работы в данной области.

2.3.3. Программная инженерия в конкретных предметных областях

Для того чтобы оценивать возможные решения с учетом различных факторов, связанных с функционированием, стоимостью, производительностью и технологичностью, инженер должен обладать опытом и образованием в соответствующей предметной области.

Инженерам необходимо принимать решения, какие из стандартных элементов могут быть использованы в осуществляемом проекте, а какие необходимо разработать с нуля. Для принятия такого рода решений они должны обладать основными знаниями в конкретной предметной области и, возможно, связанных с ней областях.

Эффективное использование специфических для предметной области методов, средств и компонент в большинстве случаев обеспечивает успешность разработок с использованием программной инженерии. Прекрасные результаты дос-

тигнуты в хорошо изученных предметных областях, где широко применяются различные стандартизированные подходы к реализации. Выпускники, специализирующиеся на программной инженерии, должны быть знакомы хотя бы с одной из прикладных предметных областей. То есть они должны понимать круг задач, которые определяют предметную область, а также общие подходы, включая стандартные компоненты (если таковые есть), используемые в производстве программного обеспечения для решения задач данной предметной области.

2.4. Профессиональная деятельность

Ключевой целью любой учебной программы в области инженерии является предоставление выпускникам знаний и начального опыта, необходимых для начала профессиональной инженерной деятельности. Как указано в главе 3, важным руководящим принципом для данного документа является «Обучение всех студентов, специализирующихся по программной инженерии, обязательно должно включать в себя практический опыт профессиональной деятельности». Содержание и смысл такого опыта рассматриваются в следующих главах, в то время как данный раздел содержит обоснование включения элементов профессиональной деятельности в учебный план по программной инженерии.

2.4.1. Обоснование

У специалистов есть определенные обязательства, которые требуют от них применения профессиональных знаний для поддержки членов общества, не имеющих таких знаний. Все характеристики инженерии, описанные в разделе 2.3.1, прямо или косвенно относятся к профессиональной инженерной деятельности. Работодатели выпускников – разработчиков программного обеспечения часто требуют того же [Denning 1992]. Каждый год Национальная ассоциация колледжей и работодателей (National Association of Colleges and Employers) проводит исследование, чтобы определить, какие качества работодатели больше всего ценят в соискателях [NACE 2003]. В 2003 году работодателей попросили оценить важность качеств и навыков кандидатов по пятибалльной шкале, где пятерка означала «чрезвычайно важно» и единица – «не важно». Наиболее востребованными характеристиками были указаны: навыки коммуникации (средний балл 4.7), честность (4.7), навыки работы в команде (4.6), навыки межличностных отношений (4.5), мотивация и инициатива (4.5), строгая этика работы (4.5).

Проблема критической зависимости общества от качества и стоимости программного обеспечения в условиях относительной незрелости программной инженерии делает вопрос профессионализма еще более важным для учебных планов по программной инженерии, чем для других инженерных программ. Выпускникам по специальности «программная инженерия» необходимо придти на рабочие места подготовленными как к решению реальных задач, так и к содействию раз-

вития дисциплины программной инженерии до более профессионального и приемлемого уровня. Как и другим специалистам в области инженерии, разработчикам программного обеспечения необходимо везде, где это уместно и допустимо, уметь находить необходимую для принятия решений количественную информацию, а также быть способными эффективно функционировать в условиях неопределенности и избегать неоправданных упрощений при моделировании.

2.4.2. Кодекс этических норм профессионала в области программной инженерии

Программная инженерия как профессия имеет определенные обязательства перед обществом. Продукты, созданные программистами, влияют на жизнь и деятельность клиентов и пользователей. Очевидно, что разработчики программного обеспечения должны действовать этично и профессионально. Преамбула к «Кодексу этических норм профессионала в области программной инженерии» [ACM 1998] формулирует это следующим образом:

> Вследствие специфики своих ролей в процессе создания программных систем инженеры по программному обеспечению имеют неограниченные возможности приносить пользу или причинять вред как самостоятельно, так и способствуя другим либо влияя на других. Инженеры должны принять на себя обязательство сделать программную инженерию полезной и уважаемой профессией, чтобы быть уверенными в том, что их работа используется во благо. Как следствие данного обязательства инженеры по программному обеспечению должны строго придерживаться «Кодекса этических норм профессионала в области программной инженерии».

Для содействия в обеспечении этически корректного и профессионального поведения преподаватели программной инженерии обязаны не только ознакомить студентов с «Кодексом», но и вовлечь их в активное обсуждение, иллюстрирующее и освещающее восемь принципов «Кодекса», а также основные дилеммы, с которыми сталкиваются профессиональные инженеры в типичных рабочих ситуациях.

2.4.3. Вопросы профессиональной деятельности в учебном плане

Учебный план может непосредственно влиять на некоторые факторы профессиональной деятельности (например, на способность работать в команде, навыки коммуникации и аналитические навыки), в то время как остальные факторы (такие как строгая рабочая этика, уверенность в собственных силах) являются предметом более тонкого влияния образования на характер индивидуума, его личные качества и зрелость. В главе 4 данного тома указаны те элементы профессиональной деятельности, которые должны быть частью любого учебного плана, а также ожидаемые результаты обучения студентов.

Главы 5 и 6 содержат рекомендации и идеи о методах включения материала по профессиональной деятельности в учебный план по программной инженерии. В частности, там представлены анализ материала, непосредственно относящегося к профессиональной деятельности (такого как технические коммуникации, этика, инженерная экономика и т.д.), и идеи моделирования рабочих ситуаций (учебные примеры, лабораторные работы, коллективно осуществляемые учебные проекты).

Существует множество различных факторов, которые существенно влияют на подготовленность студентов к профессиональной деятельности. Примерами таких факторов являются: участие в составлении учебного плана преподавателей, имеющих профессиональный опыт; опыт работы студентов в качестве практикантов; участие студентов в совместных образовательных мероприятиях; разнообразная внепрограммная деятельность, например посещение семинаров, экскурсии на предприятия и участие в профессиональных студенческих клубах и сообществах.

2.5. Предшествующая работа над учебными планами по информатике и компьютингу

В конце 1970-х годов IEEE-CS сделал попытку разработать учебный план преподавания программной инженерии, который был использован в создании множества программ подготовки магистров в США [Freeman 1976, Freeman 1978]. Эти работы сформировали общие предпосылки для начала целевой работы над учебными планами по программной инженерии. В Великобритании первые программы по подготовке бакалавров с названием «программная инженерия» начались в Королевском Колледже в 1985 году и в Шеффилдском Университете в 1988 году [Finkelstein 1993, Cowling 1998].

В конце 1980-х и начале 1990-х годов образование в области программной инженерии стимулировалось и поддерживалось работами Образовательной группы Института программной инженерии (Software Engineering Institute, SEI) в Университете Карнеги Меллон (Carnegie Mellon University, CMU). Данные работы включали: мониторинг состояния обучения программной инженерии, выпуск рекомендаций по преподаванию программной инженерии в университетах, учреждение программы по подготовке магистров в области программной инженерии в Университете Карнеги Меллон, организация и содействие семинарам преподавателей программной инженерии и публикация отдельных модулей образовательной программ [Budgen 2003, Tomayko 1999].

Институт программной инженерии инициировал и спонсировал первую Конференцию по образованию и профессиональной подготовке в области программной инженерии (Conference on Software Engineering Education & Training, CSEET), которая была проведена в 1987 году. С тех пор CSEET является регулярным форумом для встречи преподавателей программной инженерии, на котором

обсуждается широкий спектр актуальных теоретических и практических вопросов. В 1995 году в рамках своей образовательной программы SEI создал Рабочую группу по образованию и профессиональной подготовке в области программной инженерии (Working Group on Software Engineering Education and Training, WGSEET, см. http://www.sei.cmu.edu/collaborating/ed/workgroup-ed.html). Целями WGSEET являются изучение проблем, предложение вариантов решений, а также обмен информацией и зарекомендовавшими себя подходами (best practices) в рамках сообщества преподавателей программной инженерии. В 1999 году рабочая группа выпустила отчет, предлагавший рекомендации по разработке и внедрению учебных планов по преподаванию программной инженерии в университетах [Bagert 1999]. За пределами США, как в масштабе отдельных стран, так и на международном уровне, также было осуществлено большое количество мероприятий, направленных на привлечение внимания научных и деловых кругов к проблемам преподавания программной инженерии. Большинство из них проводилось в рамках более крупных событий, например Конференции по профессиональной компетенции в области программной инженерии 1996 года [Myers, 1997] или на небольших конференциях, посвященных исключительно образованию в области программной инженерии, например Рабочая конференция IFIP в Гонконге 1993 [Barta, 1993] и Международный симпозиум, проведенный в Рованиеми (Финляндия) в 1997 году [Taipale, 1997].

В 1993 году IEEE-CS и ACM основали Объединенный комитет IEEE-CS/ACM по становлению программной инженерии как профессии (IEEE-CS/ACM Joint Steering Committee for the Establishment of Software Engineering as a Profession).

Позже этот комитет был заменен Координационным комитетом по программной инженерии (Software Engineering Coordinating Committee, SWECC), который координировал одновременно три процесса: разработку «Кодекса этических норм профессионала в области программной инженерии» [ACM 1998]; деятельность в рамках Образовательного проекта по программной инженерии (SWEEP), в котором был сформирован черновой вариант аккредитационных критериев для университетских программ изучения программной инженерии [Barnes 1998]; разработку документа «Руководство к совокупности знаний по программной инженерии» (SWEBOK) [Bourque 2001]. Также существенное влияние на структуру и содержание данного документа оказали рекомендации по преподаванию информатики за 1991 год [Tucker 1991] и том CCCS [ACM 2001].

Все упомянутые выше работы повлияли на философию и содержание данного документа.

2.6. SWEBOK и другие проекты определения совокупностей знаний

Основной задачей в написании рекомендаций по преподаванию новых, недавно появившихся или динамично развивающихся дисциплин является определение и подробное изложение основного содержания дисциплины. Так как дисциплины компьютинга являются и относительно новыми и динамично развивающимися, то спецификация «совокупности знаний» чрезвычайно важна. В главе 4 указано, что учебный план преподавания программной инженерии опирается на совокупность знаний, получившую название «преподаваемый материал по программной инженерии» (SEEK, Software Engineering Education Knowledge). На организацию и содержание данного документа повлияли многие предшествующие ему работы по описанию знаний схожих дисциплин. Далее следует описание таких работ:

- SWEBOK – всестороннее описание знаний, необходимых для практической деятельности в области программной инженерии. Одной из целей проекта было «предоставить основу для разработки учебного плана ...». В поддержку достижения этой цели, SWEBOK содержит рейтинговую систему для классификации разделов знаний, основанную на классификации образовательных целей по Блуму [Bloom 1956]. Хотя SWEBOK является одним из основных ресурсов, использованных при разработке SEEK, и имеется тесная взаимосвязь проектов SWEBOK и SE2004, ряд допущений и особенностей SWEBOK существенно различают эти две работы:
  - SWEBOK охватывает знания, ожидаемые от профессионала после четырех лет работы по специальности.
  - SWEBOK преднамеренно не содержит необходимые для программных инженеров знания, которые выходят за рамки программной инженерии.
  - SE2004 рассчитан только на подготовку бакалавров по программной инженерии.
- PMBOK (Совокупность знаний по управлению проектами) [PMI 2000] предоставляет описание знаний по управлению проектами (не ограничивающееся проектами по разработке программного обеспечения). Описанные в PMBOK знания крайне важны для управления программными проектами. Кроме того, организация и стиль этого документа повлияли на подобные последующие работы в области компьютинга.
- Отчет по информационным системам 1997 года («Модель учебного плана и рекомендаций по преподаванию информационных систем») [Davis, 1997] описывает образец учебного плана преподавания информационных систем. Данный документ содержит описание совокупности знаний по информационным системам, которое включило в себя знания по программной инженерии и рекомендации по определению глубины знаний студентов (близкие к подходу, описанному Блумом).

- Документ «Компьютинг как научная дисциплина» [ACM 1989] содержит всеобъемлющее определение компьютинга, и явился основой для работы над документом Computing Curriculum 1991 и его развитием – Computing Curriculum 2001. Он выделяет девять предметных областей, которые покрывают дисциплину компьютинга, включая программную инженерию.
- Документ «Рекомендации по образованию в области программной инженерии» [Bagert 1999] (разработанный WGSEET) описывает шаблон учебного плана обучения программной инженерии в рамках университета, основанный на совокупности знаний, состоящей из четырех областей: вводной, основной, закрепления материала и поддержки.

ГЛАВА 3. Руководящие принципы

В данной главе описываются фундаментальные принципы и концепции, на которых основывалась разработка материала SE2004: руководящие принципы разработки SE2004 в целом и желаемые результаты обучения студентов в рамках программы по программной инженерии.

3.1. Принципы SE2004

На принципы разработки SE2004 существенное влияние оказали принципы, изложенные в томе CCCS (Рекомендации по преподаванию информатики, Computing Currcula 2001: Computer Science). В некоторых случаях в списке приводятся принципы из CCCS с незначительно измененной формулировкой. В других случаях мы старались раскрыть особенности природы программной инженерии, которые отличают ее от остальных дисциплин компьютинга.

[1] Компьютинг (computing) – это широкая область знаний, которая не может быть сведена к рамкам какой-либо из составляющих ее дисциплин. SE2004 концентрируется на знаниях и педагогических аспектах учебных планов по программной инженерии. Там, где возможно, SE2004 ссылается на материал, содержащийся в других томах Computing Curriculum, или частично раскрывает его. SE2004 также предлагает рекомендации по включению своего материала в учебные планы по иным дисциплинам.

[2] Программная инженерия основывается на целом ряде дисциплин. Теоретические и концептуальные основы обучения программной инженерии лежат, прежде всего, в различных областях информатики (computer science), однако для получения полноценного образования студентам необходимо быть знакомыми с рядом концепций из иных областей, таких как математика, инженерия, управление проектами и одна или же несколько конкретных предметных областей. Все студенты, изучающие программную инженерию, должны уметь интегрировать теорию и практику, понимать важность абстракции и моделирования, быть способными разбираться в новых для себя предметных областях, не связанных напрямую с компьютингом, а также понимать значимость хорошего проектирования.

[3] Быстрая эволюция программной инженерии и особенности профессиональной деятельности требуют постоянного обновления учебных планов. Профессиональные ассоциации в области программной инженерии должны поддерживать постоянный процесс пересмотра учебных программ, который позволит оперативно обновлять устаревшие составляющие учебных планов. Также, вследствие наличия у программных инженеров определенных профессиональных обязательств перед об-

ществом, важно, чтобы документ SE2004 был полезен при проведении независимых экспертиз и/или аккредитации программ обучения программной инженерии.

[4] При разработке учебных планов по информатике необходимо учитывать изменения в технологиях, методиках и приложениях, новые разработки в сфере педагогики, а также важность концепции «обучения на протяжении всей жизни» (lifelong learning). В такой быстро развивающейся области, как программная инженерия, учебные заведения должны оперативно перенимать передовые стратегии, реагируя на происходящие изменения. Университеты не должны отставать от прогресса, как в области технологий, так и в области педагогики, даже с учетом существующих ограничений в ресурсах. Кроме того, обучение программной инженерии в университете должно готовить студентов к дальнейшему обучению на протяжении всей жизни, что позволит им идти в ногу со временем и быть способными разрешать сложные проблемы будущего.

[5] SE2004 не должен ограничиваться описанием разделов знаний – необходимо также предложить набор рекомендаций по разработке отдельных курсов. Образовательные модели, описанные в SE2004, должны позволить скомпоновать разделы знаний в рационально построенные и легко реализуемые учебные модули. Четкая формулировка образовательных моделей существенно упростит взаимообмен педагогическими стратегиями и инструментами между университетами. Это также обеспечит определенной базой издателей, выпускающих учебники и другие образовательные материалы.

[6] SE2004 должен определить фундаментальные навыки и знания, которыми необходимо обладать всем выпускникам, специализирующимся на программной инженерии. По возможности SE2004 должен способствовать выявлению ключевых тем дисциплины программной инженерии и гарантировать включение соответствующего материала во все университетские учебные программы.

[7] Рекомендации по разработке учебных планов в области программной инженерии должны быть основаны на соответствующем определении совокупности знаний по программной инженерии. Описание этой совокупности знаний должно быть кратким, подходящим для использования в высшем образовании и должно опираться на наработки предыдущих исследований. Это описание должно содержать обязательный набор тем, необходимых для всех университетских программ по программной инженерии. Данный набор должен быть общепринятым в сообществе преподавателей программной инженерии. Список обязательных тем должен начинаться с материала для вводных курсов, распространяться на основной учебный план и сопровождаться материа-

лом дополнительных курсов, которые могут варьироваться в зависимости от потребностей университета, учебной программы или отдельного студента.

[8] *SE2004 должен быть полезным для всего мирового сообщества.* Несмотря на то, что учебные программы отличаются в разных странах, SE2004 должен быть полезным для преподавателей компьютинга по всему миру. Необходимо приложить все усилия к тому, чтобы данные рекомендации по составлению учебных планов учитывали национальные и культурные различия и были применимы во всем мире. Необходимо активно искать и способствовать любым возможностям привлечения к участию в составлении рекомендаций как национальных профессиональных сообществ, так и индивидуальных экспертов всех стран.

[9] *В разработку SE2004 должен быть вовлечен максимальный круг заинтересованных лиц.* Для достижения успеха к процессу создания рекомендаций по организации образования в области программной инженерии должны быть привлечены как преподаватели программной инженерии, так и представители различных заинтересованных сторон, в том числе индустрии, бизнеса и правительственных органов.

[10] *SE2004 должен включать в себя профессиональную практику в качестве неотъемлемой части учебного плана.* Профессиональная деятельность в области программной инженерии заставляет специалистов сталкиваться с широким спектром вопросов и задач, таких как разрешение проблем, менеджмент и управление, практическое применение этических и моральные норм, письменное и устное общение, работа в команде и необходимость постоянного изучения последних достижений в быстро меняющейся дисциплине.

[11] Наравне с рекомендациями высокого уровня SE2004 должен включать в себя обсуждение различных тактик и методик реализации учебных программ. Хотя для SE2004 очень важно предоставить обобщенное видение вопросов обучения программной инженерии, успех любого учебного плана существенно зависит от деталей реализации. SE2004 должен предоставить университетам советы по практическим вопросам реализации учебного плана.

3.2. Результаты обучения студентов

При составлении SE2004 в первую очередь был разработан набор результатов, ожидаемых от учебной программы. Данный набор представляет собой общий список, который может быть адаптирован к разнообразным реализациям учебных программ по программной инженерии.

По окончании университетского обучения выпускники должны:

[1] Демонстрировать владение знаниями и навыками в области программной инженерии, а также иметь профессиональные качества, необходимые для начала работы в качестве инженера по программному обеспечению.

Студентам необходимо развивать уверенность в своих возможностях посредством постоянного укрепления знаний и практики на протяжении всего периода обучения программной инженерии. В большинстве случаев знания, как и навыки, развиваются путем поэтапного подхода – различные уровни достигаются по мере углубления обучения. Также студенты должны обрести понимание и способность самостоятельно решать профессиональные вопросы, связанные с этикой профессионального поведения, экономикой и общественными потребностями.

[2] В процессе работы над программными продуктами быть способными эффективно решать поставленные перед ними задачи как индивидуально, так и в команде.

В реальной жизни студентам предстоит выполнять значительное количество проектов в одиночку, однако большинство задач требует работы в команде с другими людьми. Соответственно, студенты должны овладеть максимально полной информацией о сущности работы коллектива и ролях в команде. Они должны понимать важность таких вопросов, как дисциплинированный подход, необходимость придерживаться установленных сроков и оценка как индивидуальной, так и командной производительности.

[3] Разрешать противоречия в стоящих перед проектом целях, находя приемлемые компромиссы в рамках существующих ограничений (стоимость, время, знания, существующие системы и организации и т.п.).

Студенты должны выполнять задания, умышленно содержащие противоречивые и даже изменяющиеся требования. В таких учебных примерах должны присутствовать элементы, придающие им реалистичность. Модули учебного плана должны явно включать соответствующие темы.

[4] Проектировать решения в одной или более предметных областях, используя подходы программной инженерии, балансирующие этические, общественные, юридические и экономические интересы различных заинтересованных сторон.

На протяжении обучения студентам необходимо научиться использовать множество различных подходов к инженерному проектированию как в общем, так и к решению специфических проблем в конкретных предметных областях. Студенты должны понимать достоинства и недостатки различных доступных альтернатив и последствия выбора того или иного подхода в каждой конкретной ситуации. В предлагаемых ими проектных решениях должны адекватно учитываться этические, общественные, юридические, экономические факторы, а также вопросы безопасности.

[5] Демонстрировать понимание и способность к применению распространенных теорий, моделей и методов, которые обеспечивают современную базу для идентификации и анализа проблем, проектирования, разработки, реализации, аттестации и документирования программного обеспечения.

В этом отношении существенным является итоговый дипломный проект (Capstone project), который представляет собой крайне важную деятельность, логически завершающую обучение. Дипломный проект дает студентам возможность выполнить крупный проект и продемонстрировать умение объединять знания из различных курсов и эффективно их применять. Он позволяет студентам продемонстрировать понимание широкого спектра тем программной инженерии и способность применять приобретенные навыки для достижения желаемого эффекта. И, конечно, применить способность критически оценивать собственные действия и достижения.

[6] Демонстрировать понимание важности и способность к ведению переговоров, способность результативно работать, осуществлять руководство и эффективно общаться с заинтересованными лицами в типичных для разработки программного обеспечения ситуациях.

В программе обучения обязательно должно быть предусмотрено осуществление хотя бы одной достаточно серьезной деятельности, требующей разработки решения для некоторого заказчика. Программные инженеры должны осознавать, что им необходимо создавать программное обеспечение, прежде всего являющееся полезным. По возможности, мы должны объединить в программе обучения период производственного опыта, лекции приглашенных практикующих инженеров и даже участие во внешних конкурсах по созданию программного обеспечения. Все вместе это способствует получению более насыщенного опыта и созданию необходимой среды для подготовки высококвалифицированных специалистов по программной инженерии.

[7] Изучать новые модели, методы и технологии по мере их появления, а также осознавать необходимость постоянного профессионального роста.

По завершении программы обучения студенты должны продемонстрировать способность и стремление к самообучению на протяжении всей жизни. Данное качество достигается посредством ряда этапов на различных стадиях программы обучения. Например, на стадии разработки итогового (дипломного) проекта студенты должны быть готовы к изучению новых концепций. Но, опять же, для достижения этого необходимо на более ранних стадиях обучать студентов хорошо зарекомендовавшим себя методикам саморазвития и самообучения.

ГЛАВА 4. Обзор совокупности знаний по программной инженерии

В данной главе описывается совокупность знаний, соответствующая программе подготовки бакалавров по специальности «Программная инженерия». Систематизированные знания в данной области получили название Software Engineering Education Knowledge (SEEK – преподаваемый материал по программной инженерии).

4.1. Процесс формирования SEEK

Модель разработки документа, принятая при создании SE2004 (Software Engineering 2004), основывалась на модели построения CCCS (Computing Curricula: Computer Science). Начальный выбор областей SEEK основывался на SWEBOK и информации, полученной в результате обсуждений данной темы десятками экспертов в областях SEEK. Эксперты были разделены на группы, в среднем из семи человек, каждая из которых работала над своей областью SEEK. Перед группами была поставлена задача разработки модулей, составляющих определенные области знаний, с последующим их разбиением на темы. Для облегчения решения данной задачи группам были предоставлены ссылки на существующие работы в данной области (SWEBOK, экзаменационный курс CSDP и рекомендации по построению учебного плана SEI), а также предоставлены шаблоны документов описания модулей и тем.

После того как каждая из групп специалистов создала предварительную версию своей области из набора преподаваемых знаний, организационный комитет провел очный форум, на котором в результате обсуждения была создана предварительная версия SEEK (приложение Б содержит перечень участников). Данный очный семинар повторил успех аналогичного семинара, организованного несколько лет назад при подготовке CCCS. Когда содержание областей знаний стабилизировалось, были выделены обязательные и факультативные темы. Темам был присвоен идентификатор, определяющий один из трех уровней целей обучения согласно классификации Блума: знание, понимание, применение. Использовались не все, а только эти три уровня по той причине, что именно они соответствуют ожидаемой от бакалавров глубине знаний.

По результатам семинара был сформирован проект документа SEEK. Данный проект прошел несколько этапов рецензирования – вначале внутреннее рецензирование, а затем и рецензирование широким кругом специалистов. По завершении каждой стадии рецензирования организационный комитет изучал комментарии рецензентов для дальнейшего уточнения и улучшения содержимого SEEK.

4.2. Области знаний, модули и темы

Под словом «знание» мы понимаем термин, описывающий весь спектр содержимого дисциплины: информацию, терминологию, артефакты, данные, роли, методы, процедуры, методики, процессы и литературу. SEEK имеет иерархическую структуру, состоящую из трех уровней. На верхнем уровне иерархии находятся **области преподаваемых знаний**, представляющие собой отдельные поддисциплины программной инженерии, которые признаны важными составляющими совокупности знаний по программной инженерии (software engineering body of knowledge), необходимыми для изучения студентами. Области знаний являются элементами верхнего уровня, используемыми для систематизации, классификации и описания знаний по программной инженерии. Каждая область имеет свою аббревиатуру, например PRF (профессиональная практика).

Каждая область состоит из **модулей**, представляющих отдельную тематическую единицу области. Модули определяются добавлением двух- или трехсимвольного суффикса к идентификатору области, например PRF.com – модуль, посвященный навыкам коммуникации.

Каждый модуль разделен на несколько **тем**, являющихся низшим уровнем иерархии.

4.3. Основной материал

При формировании SEEK координационный комитет осознавал, что программная инженерия как дисциплина является сравнительно новой, развивающейся наукой и что совокупность преподаваемых знаний будет меняться в процессе ее эволюции. SEEK, разработанный и представленный в данном документе, основан на множестве предшествовавших исследований и рекомендаций. Документ был специально разработан в помощь создателям учебных планов по подготовке бакалавров по специальности «Программная инженерия» и, вследствие этого, не включает полный набор знаний, который мог бы войти в более общую совокупность знаний. Поэтому координационный комитет стремился определить основной набор знаний, состоящий из наиболее важного материала, который, как согласились эксперты в области преподавания программной инженерии, необходим любому студенту, изучающему данную область. Настаивая на необходимости выработки консенсуса в том, что входит в основной набор знаний, границах и наполнении ядра учебного плана, координационный комитет надеялся минимизировать обязательную часть учебного плана, чтобы дать учебным заведениям максимальную свободу в подборе факультативных компонент в соответствии с их потребностями.

Необходимо обратить внимание на следующие моменты, раскрывающие взаимосвязь между SEEK и конечной целью координационного комитета, состоящей в предоставлении рекомендаций к разработке учебных планов по программной инженерии.

- *Основная часть (ядро) учебного плана – это еще не весь план.* Поскольку ядро определено как минимально необходимый набор знаний, само по себе оно не является полным учебным планом для подготовки бакалавров. Каждая учебная программа должна включать в себя дополнительные модули, как относящиеся к совокупности знаний по программной инженерии, так и выходящие за ее рамки и не упоминаемые в данном документе.
- *Основные (обязательные) модули не обязательно должны быть представлены во вводных курсах, читаемых в начале обучения.* Хотя большое количество обязательных модулей действительно являются вводными, некоторые из них могут быть включены в образовательный процесс только после того, как студенты получат достаточные базовые знания в данной области. Например, темы, связанные с управлением проектами, выявлением требований, абстрактным высокоуровневым моделированием и т.п., могут потребовать такого уровня знаний и владения материалом, каким студенты начальных курсов еще не обладают. Конечно, помимо обязательного материала, вводные курсы могут содержать дополнительные темы[1]. Классификация материала как основного означает обязательное его включение в учебный план, но ничего не говорит о том, когда (на каком этапе) он должен в нем появиться.

4.4. Единица времени

SEEK должен определить шкалу, которая бы установила стандарты измерения времени, необходимого для изучения определенного модуля. Выбор такой метрики был довольно сложным, так как в мире отсутствует общепринятая система измерения длительности курса. Для соблюдения соответствия с более ранними рекомендациями по составлению учебных планов, рабочая группа решила определять время в часах. **Час** соответствует фактическому аудиторному (in-class) времени, называемому в данном документе также временем контакта (contact hours), которое требуется для преподавания материала в традиционном лекционном формате. Однако, чтобы исключить любые возможные ошибки, необходимо подчеркнуть следующие замечания относительно использования лекционных часов в качестве меры измерения.

- *Координационный комитет не предполагает навязывание лекционного формата.* Несмотря на то, что используемая нами мера измерений имеет общие корни с классическим лекционным форматом, координационный комитет считает, что существуют другие методы (особенно принимая во внимание последние достижения в технологиях обучения), которые, по меньшей мере, будут столь же эффективны. К некоторым методам применение понятия учебных часов может быть затруднено. Но даже и в этом случае временные единицы измерения должны как минимум слу-

[1] Материалы, являющиеся частью программы подготовки бакалавров и не входящие в основной набор знаний, считаются **факультативными**.

жить в качестве сравнительной характеристики, то есть 5-часовой модуль займет приблизительно в пять раз больше времени, чем часовой, независимо от стиля обучения.

- *Приведенное количество часов не включает время, проведенное вне аудитории.* Время, отведенное на изучение модуля, не включает времени подготовки преподавателя, а также времени, которые студенты проводят вне аудитории. Общая рекомендация такова: время на работу студентов вне аудитории приблизительно в три раза больше времени внутри ее (3 часа в аудитории и 9 на самостоятельную работу).
- *Количество часов, указанное для модуля, рассчитано на минимальный уровень изложения.* Временные рамки, указанные для каждого модуля, должны расцениваться как минимальное количество времени, необходимое для достижения учебных целей данного модуля. Всегда допустимо выделять на изучение модуля больше времени, чем указано в обязательном минимуме.

4.5. Связь SEEK и учебного плана

SEEK не представляет собой весь объем учебного плана – он дает основу для проектирования, организации и преподавания модулей знаний, составляющих учебный план по программной инженерии. Остальные главы SE2004 содержат рекомендации относительно использования SEEK для разработок учебных планов. Ни организация, ни содержание областей и модулей знаний SEEK не должны оказывать влияния на способ их преподавания. К примеру, SEEK не требует сохранения последовательности областей знаний SEEK в разрабатываемом плане (сначала CMP, затем FND, потом PRF и т.д.). Здесь также нет схемы объединения тем и модулей SEEK в модули учебного плана. Разработчики SEEK также не преследовали цель выбрать определенную методологию разработки учебного курса (водопадный метод, инкрементный, циклический и др.).

4.6. Выбор областей знаний

Отправной точкой для определения областей знаний послужило руководство SWEBOK. Так как координационный комитет SE2004 и участники рабочих групп по SEEK были уверены в необходимости акцентирования внимания на академическом характере дисциплины, области знаний, представляющие теоретические и научные основы разработки программных продуктов, выросли в итоге до половины объема основных (обязательных) знаний. Это вызвало необходимость повторного анализа – действительно ли были достигнуты изначально сформулированные цели. Откорректированный в результате этого процесса набор областей знаний стал достаточно сбалансирован. Полученный в итоге документ описывает фундаментальные принципы, знания и практические методики, лежащие в основе дисциплины программной инженерии, в форме, приемлемой для подготовки бакалавров.

4.7. Области преподаваемого материала по программной инженерии

В данном разделе описаны десять областей знаний, составляющих SEEK:

1. Основы компьютинга (CMP).
2. Основы математики и инженерии (FND).
3. Профессиональная практика (PRF).
4. Моделирование и анализ программного обеспечения (MAA).
5. Проектирование программного обеспечения (DES).
6. Верификация и аттестация программного обеспечения (VAV).
7. Эволюция программного обеспечения (EVL).
8. Процессы разработки программного обеспечения (PRO).
9. Качество программного обеспечения (QUA).
10. Управление программными проектами (MGT).

В области знаний по программной инженерии не включены материалы по непрерывной математике и естественным наукам – потребность в этих знаниях обсуждается в других частях документа SE2004. Для каждой области знаний приводится ее краткое описание, а также таблица, определяющая разбиение области на отдельные модули и темы. Для каждого модуля области знаний указано рекомендуемое количество аудиторных часов. Для каждой темы определен уровень по классификации Блума, показывающий, каким уровнем владения данной темой должен обладать выпускник, а также значимость темы, показывающая, является ли данная тема необходимой, желаемой либо факультативной по отношению к основному набору знаний по программной инженерии. В таблице 1 перечислены все области SEEK с указанием их модулей и минимального количества часов, необходимого для их изучения.

Атрибуты классификации Блума определяются с помощью одного из символов k, c или a и означают следующее:

- Знание (k) – Запоминание ранее пройденного материала. Достижение этого уровня проверяется по способности вспомнить ту или иную информацию (например, дату, событие, место, знание основных идей, знание предметной области).
- Понимание (c) – Осмысление информации и значения изучаемого материала. Например, способность использовать полученные знания в другом контексте, умение интерпретировать факты, проводить сравнение, находить различия, упорядочивать, группировать, делать выводы о причинах, предугадывать последствия и т.д.
- Применение (a) – Умение использовать изученный материал в новых и конкретных ситуациях. Например, использование информации, методов, концепций и теоретических подходов для решения проблем, требующих полученных навыков и знаний.

Значимость темы по отношению к основному набору знаний по программной инженерии представлена следующим образом:

- Обязательная (E) – тема входит в основной набор знаний по программной инженерии.
- Желаемая (D) – тема не входит в основной набор знаний по программной инженерии, но по возможности ее следует включать в конкретные учебные планы; в противном случае тему следует предлагать в качестве курса по выбору.
- Факультативная (O) – тему следует предлагать только в качестве курсов по выбору.

Таблица 1. Области и модули знаний SEEK*

Области знаний и темы	**Наименование области или модуля знаний**	**Часы**	**Области знаний и темы**	**Наименование области или модуля знаний**	**Часы**
CMP	Основы компьютинга	172	VAV	Верификация и аттестация программного обеспечения	42
CMP.cf	Основы информатики	140	VAV.fnd	Терминология и основы верификации и аттестации	5
CMP.ct	Технологии разработки	20	VAV.rev	Рецензии кода	6
CMP.tl	Средства разработки	4	VAV.tst	Тестирование	21
CMP.fm	Формальные методы разработки программного обеспечения	8	VAV.hct	Тестирование и оценка пользовательского интерфейса	6
			VAV.par	Анализ проблем и создание отчетов	4
FND	Основы математики и инженерии	89	EVL	Эволюция программного обеспечения	10
FND.mf	Основы математики	56	EVL.pro	Процессы эволюции	6
FND.ef	Инженерные основы программного обеспечения	23	EVL.ac	Эволюционное развитие программного обеспечения	4
FND.ec	Инженерная экономика программного обеспечения	10			
PRF	Профессиональная практика	35	PRO	Процессы разработки программного обеспечения	13
PRF.psy	Групповая динамика /психология	5	PRO.con	Концепции программных процессов	3

PRF.com	Навыки коммуникации (характерные для программной инженерии)	10	PRO.imp	Реализация программных процессов	10
PRF.pr	Профессионализм	20			
MAA	Моделирование и анализ программного обеспечения	53	QUA	Качество программного обеспечения	16
MAA.md	Основы моделирования	19	QUA.cc	Концепции и культура качества программного обеспечения	2
MAA.tm	Типы моделей	12	QUA.std	Стандарты качества программного обеспечения	2
MAA.af	Основы анализа	6	QUA.pro	Процессы обеспечения качества программного обеспечения	4
MAA.rfd	Основы управления требованиями	3	QUA.pca	Обеспечение качества процесса	4
MAA.er	Выявление требований	4	QUA.pda	Обеспечение качества продукта	4
MAA.rsd	Спецификация и документирование требований	6			
MAA.rv	Аттестация требований	3			
DES	Проектирование программного обеспечения	45	MGT	Управление программными проектами	19
DES.con	Концепции проектирования	3	MGT.con	Концепции менеджмента	2
DES.str	Стратегии проектирования	6	MGT.pp	Планирование проектов	6
DES.ar	Архитектурное проектирование	9	MGT.per	Организация и управление персоналом	2
DES.hci	Проектирование человеко-машинного интерфейса	12	MGT.ctl	Отслеживание выполнения проектов	4
DES.dd	Детальное проектирование	12	MGT.cm	Управление конфигурацией программного обеспечения	5
DES.ste	Оценка и средства поддержки проектирования	3			

* Раздел 4.18 (Системные и прикладные специальности) включает дополнительный материал, не относящийся к основному набору тем, но позволяющий расширить основной набор знаний и предусматривающий специализацию.

4.8. Основы компьютинга

Описание

Основы компьютинга включают в себя основы информатики, необходимые для проектирования и разработки программных продуктов. Данная область знаний включает в себя также знания о трансформации проекта в реализацию, используемых при этом средствах и о формальных методах создания программного обеспечения.

Модули и темы

Обозначение	Наименование области или модуля знаний	k, c, a	E, D, O	Часы	Связанные темы
CMP	Основы компьютинга			172	
CMP.cf	Основы информатики			140	
CMP.cf.1	Основы программирования (CCCS PF1-PF5) (управление и данные, типизация, рекурсия)	a	E		CMP.ct.1, CMP.fm.5., MAA.cc.1
CMP.cf.2	Алгоритмы, структуры и представление данных. Сложность (CCCS AL1-AL5)	a	E		CMP.ct.1, CMP.fm.5, MAA.cc.1
CMP.cf.3	Методы решения задач	a	E		CMP.cf.1
CMP.cf.4	Использование и поддержка абстракции (инкапсуляция, иерархия и др.)		a	E	MAA.md.1
CMP.cf.5	Архитектура ЭВМ (CCCS AR1-AR5)	c	E		
CMP.cf.6	Базовые концепции систем	c	E		MAA.rfd.7
CMP.cf.7	Человеческий фактор – пользователи (ввод/вывод, сообщения об ошибках, устойчивость системы)	c	E		DES.hci
CMP.cf.8	Человеческий фактор – разработчики (комментарии, структура, читаемость)	c	E		CMP.cf.1
CMP.cf.9	Основы языков программирования (ключевые понятия, CCCS PL1-PL6)	a	E		CMP.ct.3, CMP.ct.4
CMP.cf.10	Основы операционных систем (ключевые понятия, CCCS OS1-OS6)	c	E		CMP.ct.10, CMP.ct.15
CMP.cf.11	Основы баз данных	c	E		DES.con.2
CMP.cf.12	Основы сетевых технологий	c	E		
CMP.cf.13	Семантика языков программирования		D		

CMP.ct	Технологии разработки программного обеспечения			20	
CMP.ct.1	Проектирование и использование API	a	E		DES.dd.4
CMP.ct.2	Библиотеки и повторное использование кода	a	E		CMP.cf.1
CMP.ct.3	Аспекты исполнения объектно-ориентированных программ (полиморфизм, динамическое связывание и т.п.)	a	E		CMP.cf.1, 9 DES.str.2
CMP.ct.4	Параметрический полиморфизм	a	E		CMP.cf.1
CMP.ct.5	Утверждения (assertions), проектирование по контракту (design by contract), защитное программирование (defensive programming)	a	E		MAA.md.2
CMP.ct.6	Обработка ошибок, обработка исключений, отказоустойчивость (fault tolerance)	a	E		DES.con.2, VAV.tst.2, VAV.tst.9
CMP.ct.7	Технологии, основанные на состояниях и табличных методах	c	E		FND.mf.7, MAA.tm.2, CMP.cf.10
CMP.ct.8	Конфигурирование системы в процессе исполнения и интернационализация	a	E		DES.hci.6
CMP.ct.9	Обработка данных на основе грамматик (синтаксический анализ)	a	E		FND.mf.8
CMP.ct.10	Базовые конструкции параллелизма (семафоры, мониторы и т.д.)	a	E		CMP.cf.10
CMP.ct.11	Промежуточное программное обеспечение (компоненты и контейнеры)	c	E		DES.dd.3,5
CMP.ct.12	Методы разработки распределенных программных систем	a	E		CMP.cf.2
CMP.ct.13	Разработка гетерогенных (программных и аппаратных) систем, совместная разработка программно-аппаратных комплексов (codesign)	c	E		DES.ar.3
CMP.ct.14	Анализ и улучшение производительности	k	E		FND.ef.4, DES.con.6, CMP.tl.4, VAV.fnd.4
CMP.ct.15	Платформенные стандарты (Posix и т.п.)		D		
CMP.ct.16	Программирование с ориентацией на тестирование (test-first programming)		D		VAV.tst.1
CMP.tl	Средства разработки			4	DES.ste.1
CMP.tl.1	Среды разработки	a	E		
CMP.tl.2	Среды проектирования графического интерфейса пользователя (GUI builders)	c	E		DES.hci

CMP.tl.3	Средства модульного тестирования	c	E		VAV.tst.1
CMP.tl.4	Предметно-ориентированные языки (скриптовые языки, языки, ориентированные на заданную предметную область, языки разметки, макросы и т.п.)	c	E		
CMP.tl.5	Средства профилирования, анализа производительности и построения срезов программ		D		CMP.ct.14
CMP.fm	Формальные методы разработки программного обеспечения			8	DES.dd.9, MAA.af.6, EVO.ac.7
CMP.fm.1	Применение абстрактных машин (например, SDL, Paisley)	k	E		
CMP.fm.2 MAA.md.3,	Применение языков и методов формальных спецификаций (ASM, B, CSP, VDM, Z и др.)	a	E		MAA.rsd.3
CMP.fm.3	Автоматическая генерация кода по спецификации	k	E		
CMP.fm.4	Вывод программ	c	E		
CMP.fm.5	Анализ реализаций-кандидатов	c	E		MAA.cf.2
CMP.fm.6	Отображение спецификации на различные реализации	k	E		
CMP.fm.7	Уточнение (refinement) программного обеспечения	c	E		
CMP.fm.8	Доказательства корректности программного обеспечения		D		FND.mf.3

4.9. Основы математики и инженерии

Описание

Математические и инженерные основы программной инженерии обеспечивают теоретическую и научную базу для разработки программных продуктов с желаемыми свойствами. Эти основы помогают дать точное описание продуктов программной инженерии. Они предоставляют математические методы для моделирования и позволяют делать умозаключения о продуктах и их взаимосвязях, а также обеспечивают базу для предсказуемого процесса проектирования. Центральной темой является инженерное проектирование, т.е. итерационный процесс принятия решений, в котором компьютинг, математика и инженерные науки используются для организации эффективного использования доступных ресурсов с целью достижения поставленной цели.

Модули и темы

Обозначение	Наименование области или модуля знаний	k, c, a	E, D, O	Часы	Связанные темы
FND	Основы математики и инженерии			89	
FND.mf	Основы математики*			56	
FND.mf.1	Функции, отношения и множества (CCCS DS1)	a	E		
FND.mf.2	Основы логики (высказывания и предикаты) (CCCS DS2)	a	E		MAA.md.2, MAA.md.3
FND.mf.3	Методы доказательства (прямое, от противного, индуктивное) (CCCS DS3)	a	E		CMP.fm.8
FND.mf.4	Основы вычислений (CCCS DS4)	a	E		
FND.mf.5	Графы и деревья (CCCS DS5)	a	E		CMP.cf.2
FND.mf.6	Дискретная вероятность (CCCS DS6)	a	E		FND.ef.2
FND.mf.7	Конечные автоматы, регулярные выражения	c	E		CMP.ct.7, MAA.tm.2
FND.mf.8	Грамматики	c	E		CMP.ct.9
FND.mf.9	Вычислительная точность, погрешность и ошибки	c	E		
FND.mf.10	Теория чисел		D		
FND.mf.11	Алгебраические структуры		O		
FND.ef	Инженерные основы программного обеспечения	23			
FND.ef.1	Эмпирические и экспериментальные методы (например, компьютерные методы измерения загруженности процессора и использования памяти)	c	E		VAV.fnd.4, VAV.hct.6
FND.ef.2	Статистический анализ (включая проверку простых гипотез, оценки, регрессию и корреляцию)	a	E		FND.mf.6
FND.ef.3	Измерение и метрики	k	E		PRO.con.5, PRO.imp.4
FND.ef.4	Разработка систем (защищенность, безопасность, производительность, эффекты масштабирования, взаимодействие различных функций и т.д.)	k	E		MAA.af.4, DES.con.6, VAV.fnd.4, VAV.tst.9

FND.ef.5	Инженерное проектирование (формулировка проблемы, поиск различных решений, оценка осуществимости)		c	E	FND.ec.3, MAA.af.1
FND.ef.6	Теория измерений (в частности, критерии правильности измерений)	c	E		
FND.ef.7	Инженерные принципы в других инженерных дисциплинах (сопротивление материалов, принципы цифровых систем, логическое проектирование, основы термодинамики и т.д.)		O		
FND.ec	Инженерная экономика программного обеспечения		10		PRF.pr.6
FND.ec.1	Измерение стоимости на протяжении жизненного цикла программного обеспечения	k	E		
FND.ec.2	Определение системных целей (вовлечение заказчика в проектирование, обеспечение выигрыша для всех участвующих сторон (stakeholder win-win), внедрение функций качества (quality function deployment), прототипирование и др.)	c	E		PRF.psy.4, MAA.er.2
FND.ec.3	Сравнение решений по критерию «цена-качество» (преимущества от внедрения, анализ компромиссов, анализ стоимости, возврат инвестиций и т.д.)	c	E		DES.con.7, MAA.af.4, MGT.pp.4
FND.ec.4	Управление стоимостью системы (расстановка приоритетов, управление рисками, управление затратами и др.)	k	E		MAA.af.4, MGT.pp.6

*Темы 1-6 соответствуют разделу «Дискретные структуры» тома «Рекомендации по преподаванию информатики» (Computing Curricula 2001: Computer Science).

4.10. Профессиональная практика

Описание

Профессиональная практика изучает знания, навыки и отношение к делу, которыми должны обладать выпускники для того, чтобы заниматься программной инженерией профессионально, ответственно и соблюдая этические принципы. Изучение профессиональной практики включает в себя вопросы технических коммуникаций, групповой динамики и психологии, а также социальной и профессиональной ответственности.

Модули и темы

Обозначение	Наименование области или модуля знаний	k, c, a	E, D, O	Часы	Связанные темы
PRF	Профессиональная практика			35	
PRF.psy	Групповая динамика / психология			5	
PRF.psy.1	Психология работы в команде	a	E		
PRF.psy.2	Индивидуальные познавательные способности (в том числе ограничения)	k	E		DES.hci.10
PRF.psy.3	Когнитивная сложность проблем	k	E		MAA.rfd.8
PRF.psy.4	Взаимодействие со всеми заинтересованными сторонами проекта	c	E		FND.ec.2
PRF.psy.5	Деятельность в условиях неопределенности и двусмысленности	k	E		
PRF.psy.6	Работа в мультикультурных средах	k	E		
PRF.com	Навыки коммуникации (специфичные для программной инженерии)			10	
PRF.com.1	Чтение, понимание и выделение главной идеи прочитанного (исходный код, документация)	a	E		MAA.rsd.1
PRF.com.2	Навыки делового письма (назначения, отчеты, оценки, обоснования и т.п.)	a	E		
PRF.com.3	Общение в команде или группе (устное, письменное, по электронной почте и т.д.)	a	E		MGT.per
PRF.com.4	Презентационные навыки	a	E		
PRF.pr	Профессионализм			20	
PRF.pr.1	Аккредитация, сертификация и лицензирование	k	E		
PRF.pr.2	Кодексы этики и профессионального поведения	c	E		
PRF.pr.3	Социальные, юридические, исторические и профессиональные вопросы	c	E		
PRF.pr.4	Природа и значимость профессиональных сообществ	k	E		
PRF.pr.5	Природа и значимость стандартов в области программной инженерии	k	E		MAA.rsd.1, CMP.ct.14, PRO.imp.3, 7, QUA.std
PRF.pr.6	Экономическое влияние программного обеспечения	c	E		FND.ec
PRF.pr.7	Контракты и наем на работу	k	E		

4.11. Моделирование и анализ программного обеспечения

Описание

Моделирование и анализ могут рассматриваться как основные концепции в любой инженерной дисциплине, так как они необходимы для документирования и оценки проектных решений и альтернатив. Моделирование и анализ в первую очередь применяются к анализу, спецификации и аттестации требований. Требования представляют собой реальные потребности пользователей, клиентов и других заинтересованных лиц, интересы которых так или иначе затрагиваются системой. Создание требований включает анализ возможности создания планируемой системы, выявление и анализ потребностей заинтересованных лиц, четкое описание того, что система должна делать и что находится за рамками системы, каковы ограничения системы по ее эксплуатации и реализации, а также аттестацию данного описания или спецификации заинтересованными лицами.

Модули и темы

Обозначение	Наименование области или модуля знаний	k, c a	E, D, O	Часы	Связанные темы
MAA	Моделирование и анализ программного обеспечения			53	
MAA.md	Основы моделирования			19	PRO.con.3, QUA.pro.1, QUA.pda.3
MAA.md.1	Принципы моделирования (декомпозиция, абстракция, обобщение, проекции/представления, ясность записи (explicitness), использование формальных подходов и др.)	a	E		CMP.cf.4
MAA.md.2	Пред- и постусловия, инварианты	c	E		CMP.ct.5
MAA.md.3	Введение в математические модели и языки написания спецификаций (Z, VDM и т.д.)	c	E		MAA.rsd.3, CMP.fm.2
MAA.md.4	Свойства языков моделирования	k	E		
MAA.md.5	Синтаксис и семантика (понимание представлений модели)	c	E		CMP.cf.9
MAA.md.6	Ясность записи (explicitness – не делайте никаких предположений или явно сформулируйте все предположения)	k	E		

MAA.tm	Типы моделей			12	MAA.md
MAA.tm.1	Информационное моделирование (например, моделирование «сущность-связь», диаграммы классов и т.д.)	a	E		MAA.rsd.3, DES.dd.5
MAA.tm.2	Поведенческое моделирование (структурный анализ, диаграммы состояний, анализ вариантов использования, диаграммы взаимодействия, варианты отказов и анализ эффектов, анализ дерева ошибок и т.д.)	a	E		FND.mf.7, MAA.er.2, MAA.rsd.3, DES.dd.5
MAA.tm.3	Структурное моделирование (архитектурное и др.)	c	E		MAA.rfd.7
MAA.tm.4	Моделирование предметной области (например, инженерия предметной области)	k	E		
MAA.tm.5	Функциональное моделирование (например, диаграммы компонентов)	c	E		
MAA.tm.6	Моделирование работы предприятия (бизнес-процессы, организации, цели и т.д.)		D		
MAA.tm.7	Моделирование встроенных систем (планируемость в режиме реального времени, анализ внешних интерфейсов и т.д.)		D		
MAA.tm.8	Анализ взаимодействия требований (взаимодействие различных функций, дом качества (house of quality), анализ точек зрения и т.д.)		D		
MAA.tm.9	Шаблоны анализа (фреймы задач, повторное использование спецификаций и т.д.)		D		
MAA.af	Основы анализа			6	
MAA.af.1	Анализ формальной правильности (well-formedness) (например, проверка полноты, согласованности, устойчивости и т.д.)	a	E		
MAA.af.2	Анализ корректности (статический анализ, имитационное моделирование, проверка модели и т.д.)	a	E		
MAA.af.3	Анализ качества (нефункциональных) требований (защищенность, безопасность, удобство использования, производительность, анализ причин (root cause analysis) и т.д.)	a	E		FND.ef.4, QUA.pda, DES.con.6, VAV.fnd.4, VAV.tst.9, VAV.hct, EVO.ac.4
MAA.af.4	Расстановка приоритетов, анализ компромиссных решений, анализ рисков и анализ последствий (impact analysis)	c	E		FND.ec.3,4 QUA.pda.4

MAA.af.5	Отслеживаемость	c	E		DES.ar.4, EVO.pro.2
MAA.af.6	Формальный анализ	k	E		CMP.fm
MAA.rfd	Основы управления требованиями			3	
MAA.rfd.1	Определение понятия «требование» (например, к продукту или проекту, ограничения, границы системы, внешние и внутренние требования)	c	E		
MAA.rfd.2	Процесс работы с требованиями	c	E		PRO.con.3
MAA.rfd.3	Слои/уровни требований (потребности, цели, пользовательские требования, системные требования, требования к программному обеспечению и т.д.)	c	E		MAA.rsd
MAA.rfd.4	Характеристики требований (проверяемость, недвусмысленность, согласованность, корректность, отслеживаемость, приоритетность и т.д.)	c	E		MAA.af.5
MAA.rfd.5	Управление изменениями требований	c	E		MGT.ctl.1
MAA.rfd.6	Управление требованиями (сохранение согласованности, планирование выпусков, повторное использование и т.д.)	k	E		CMP.ct.3
MAA.rfd.7	Взаимодействие между требованиями и архитектурой	k	E		MAA.tm.3, DES.ar.4, EVO.pro.2
MAA.rfd.8	Взаимосвязь между требованиями и системной инженерией, проектированием, ориентированным на человека (human-centered design) и т.п.		D		CMP.cf.6
MAA.rfd.9	«Плохие» (wicked) проблемы (например, плохо структурированные проблемы, проблемы с множественными решениями и т.д.)		D		PRF.psy.3
MAA.rfd.10	Использование «коробочных» программных продуктов (COTS) как ограничение		D		
MAA.er	Выявление требований			4	
MAA.er.1	Источники для выявления требований (заинтересованные лица, эксперты в предметной области, эксплуатационная и организационная среды)	c	E		PRF.psy.4
MAA.er.2	Методы выявления требований (собеседования, заполнение опросных листов/анкет, прототипы, варианты использования, наблюдение, вовлекающие методы и т.д.)	c	E		FND.ec.2, MAA.er.1, PRF.psy.5

MAA.er.3	Продвинутые методы (этнография, выявление знаний и т.д.)		O		
MAA.rsd	Спецификация и документирование требований	6			
MAA.rsd.1	Основы документирования требований (типы, целевые аудитории, структура, качество, атрибуты, стандарты и т.д.)	k	E		PRF.pr.5
MAA.rsd.2	Спецификация требований к программному обеспечению	a	E		
MAA.rsd.3	Языки написания спецификаций (структурированный естественный язык, UML, формализованные языки, такие как Z,VDM,SCR,RSML и т.д.)	k	E		MAA.md.3, CMP.fm.2
MAA.rv	Аттестация требований			3	
MAA.rv.1	Рецензирование и инспекции	a	E		VAV.rev
MAA.rv.2	Прототипирование для проверки требований (создание обобщенного прототипа – summative prototyping)	k	E		
MAA.rv.3	Проектирование приемочных тестов	c	E		VAV.tst.8
MAA.rv.4	Аттестация атрибутов качества продукции	c	E		QUA.cc.5
MAA.rv.5	Формальный анализ требований		D		MAA.af.1

4.12. Проектирование программного обеспечения

Описание

Проектирование программного обеспечения занимается проблемами, методами, стратегиями, представлениями и шаблонами, используемыми для определения способа реализации компоненты или системы. Проектирование должно соответствовать функциональным требованиям в пределах ограничений, накладываемых другими требованиями, такими как ресурсы, производительность, надежность и безопасность. Данная область также включает в себя спецификацию внутренних интерфейсов между компонентами программного обеспечения, архитектурное проектирование, проектирование данных, проектирование пользовательского интерфейса, средства проектирования и оценку проектирования.

Модули и темы

Обозначение	Наименование области или модуля знаний	k, c, a	E, D, O	Часы	Связанные темы
DES	Проектирование программного обеспечения			45	
DES.con	Концепции проектирования			3	
DES.con.1	Определение проектирования	c	E		
DES.con.2	Основные вопросы проектирования (постоянные данные (persistent data), управление памятью, исключительные ситуации и т.д.)	c	E		CMP.ct.6, VAV.tst.2, CMP.cf.11
DES.con.3	Контекст проектирования в рамках нескольких циклов разработки программного обеспечения	k	E		
DES.con.4	Принципы проектирования (скрытие информации, сцепление (cohesion) и связность (coupling))	a	E		
DES.con.5	Взаимосвязь между проектированием и требованиями	c	E		DES.ar.4
DES.con.6	Проектирование атрибутов качества (надежность, удобство использования, сопровождаемость,производительность, тестируемость, безопасность, отказоустойчивость и т.п.)	k	E		FND.ef.4, MAA.tm.4, DES.ar.2, CMP.ct.14, VAV.fnd.4
DES.con.7	Компромиссные решения при проектировании	k	E		FND.ec.3, DES.ar.2, DES.ev
DES.con.8	Архитектурные стили, шаблоны, повторное использование	c	E		DES.ar, DES.dd.2, CMP.ct.3
DES.str	Стратегии проектирования			6	
DES.str.1	Функционально-ориентированное проектирование	a,c	E		
DES.str.2	Объектно-ориентированное проектирование	c,a	E		CMP.cf.9, DES.dd.5, CMP.ct.4
DES.str.3	Проектирование от данных		D		
DES.str.4	Аспектно-ориентированное проектирование		O		

DES.ar	Архитектурное проектирование			9	
DES.ar.1	Архитектурные стили (например,«каналы-«каналы-и-фильтры» (pipe-and-filter), многоуровневый, транзакционный, одноранговый (peer-to-peer), «публикация-и-подписка», основанный на событиях, клиент-серверный и т.д.)	a	E		DES.con.8
DES.ar.2	Архитектурные компромиссы между различными атрибутами	a	E		FND.ec.3
DES.ar.3	Вопросы аппаратного обеспечения в программной архитектуре	k	E		CMP.ct.13
DES.ar.4	Отслеживаемость требований в архитектуре	k	E		MAA.af.5, DES.con.5, EVO.pro.2
DES.ar.5	Архитектуры для заданной предметной области (domain-specific architectures) и линейки продуктов (product lines)	k	E		
DES.ar.6	Архитектурные нотации (например, структурные точки зрения (architectural structure viewpoints) и представления (representations), компонентные диаграммы и т.д.)	c	E		MAA.tm
DES.hci	Проектирование человеко-машинного интерфейса			12	CMP.cf.7, VAV.hct, CMP.ct.2
DES.hci.1	Общие принципы проектирования человеко-машинного интерфейса	a	E		
DES.hci.2	Использование режимов и навигации	a	E		
DES.hci.3	Методы кодирования и визуальное проектирование (например, цвета, пиктограммы, шрифты, и т.д.)	c	E		
DES.hci.4	Время отклика и обратная связь	a	E		
DES.hci.5	Модальности проектирования (например, управление через меню, формы, запрос-ответ и т.д.)	a	E		
DES.hci.6	Локализация и интернационализация	c	E		CMP.ct.8
DES.hci.7	Методы проектирования человеко-машинного интерфейса	c	E		
DES.hci.8	Мультимедиа (например, методы ввода-вывода, голос, естественный язык, web-страница, звук и т.д.)		D		
DES.hci.9	Метафоры и концептуальные модели		D		
DES.hci.10	Психология человеко-машинного интерфейса		D		PRF.psy.2

DES.dd	Детальное проектирование			12	
DES.dd.1	Конкретный метод проектирования (например, SSA/SD, JSD, OOD и т.д.)	a	E		
DES.dd.2	Шаблоны проектирования	a	E		DES.con.8
DES.dd.3	Компонентное проектирование	a	E		CMP.ct.11
DES.dd.4	Проектирование интерфейса компонент и системы	a	E		CMP.ct.2
DES.dd.5	Нотации проектирования (диаграммы классов и объектов, UML, диаграммы состояний)	c	E		MAA.tm
DES.ste	Оценка и средства поддержки проектирования			3	
DES.ste.1	Средства поддержки проектирования (архитектурные, статический анализ, динамическая оценка и т.д.)	a	E		CMP.ct
DES.ste.2	Измерение атрибутов проектирования (сцепление, связность, разделение интересов (separation of concerns), скрытие информации)	k	E		
DES.ste,3	Метрики проектирования (архитектурные факторы, интерпретация, часто используемые метрики и т.д.)	a	E		
DES.ste.4	Формальный анализ проектирования		O		MAA.af.2

4.13. Верификация и аттестация программного обеспечения

Описание

Верификация и аттестация программного обеспечения используют как статические, так и динамические методы проверки системы для обеспечения соответствия реализованного программного продукта исходной спецификации и ожиданиям заинтересованных сторон.

Статические методы связаны с анализом и проверкой представлений системы на всех этапах жизненного цикла программного обеспечения, тогда как динамические методы изучают только саму реализованную систему.

Модули и темы

Обозначение	Наименование области или модуля знаний	k, c, a	E, D, O	Часы	Связанные темы
VAV	Верификация и аттестация программного обеспечения			42	
VAV.fnd	Терминология и основы верификации и аттестации программного обеспечения			5	
VAV.fnd.1	Задачи и ограничения верификации и аттестации	k	E		
VAV.fnd.2	Планирование верификации и аттестации	k	E		
VAV.fnd.3	Документирование стратегии верификации и аттестации, включая тесты и другие артефакты	a	E		
VAV.fnd.4	Метрики и измерения (например, надежность, удобство использования, производительность и т.п.)	k	E		FND.ef.4, MAA.af.2, DES.con.6, CMP.ct.14, PRO.con.4
VAV.fnd.5	Использование верификации и аттестации на различных этапах жизненного цикла	k	E		
VAV.rev	Рецензирование кода (reviews)			6	MAA.rv.1
VAV.rev.1	Проверка за столом (desk checking)	a	E		
VAV.rev.2	Сквозные просмотры текста программ (walkthroughs)	a	E		
VAV.rev.3	Инспекции	a	E		VAV.hct.2,3
VAV.tst	Тестирование			21	MAA.rfd.4, DES.con.6, CMP.ct.15
VAV.tst.1 CMP.ct.3	Модульное тестирование	a	E		CMP.ct.15,
VAV.tst..2	Обработка исключений (написание тестовых сценариев, вызывающих обработку исключений, грамотная обработка исключений)	a	E		DES.con.2, CMP.ct.6
VAV.tst.3	Анализ покрытия и структурное тестирование (оператор, ветвление, базовый путь, мультиусловие, поток данных и т.д.)	a	E		
VAV.tst.4	Функциональное тестирование методом «черного ящика»	a	E		
VAV.tst.5	Интеграционное тестирование	c	E		

VAV.tst .6	Разработка тестовых сценариев на основе вариантов использования (use case) и пользовательских историй (customer stories)	a	E		MAA.tm.2
VAV.tst.7	Эксплуатационное тестирование (operation testing) на основе профилей использования	k	E		
VAV.tst.8	Системные и приемочные тесты (acceptance testing)	a	E		MAA.rv.4
VAV.tst.9	Тестирование атрибутов качества (удобство использования, безопасность, доступность (accessibility), совместимость)	a	E		MAA.af.3, MAA.rv.6, VAV.hct, QUA.cc.5
VAV.tst.10	Регрессионное тестирование	c	E		
VAV.tst.11	Инструментальные средства тестирования	a	E		CMP.ct.3
VAV.tst.12	Процесс внедрения в эксплуатацию		D		
VAV.hct	Тестирование и оценка человеко-машинного интерфейса			6	DES.hci, VAV.tst.9
VAV.hct.1	Разнообразие аспектов полезности и удобства использования	k	E		MAA.af.3
VAV.hct.2	Эвристическая оценка	a	E		VAV.rev.3
VAV.hct.3	Когнитивный сквозной просмотр текстов	c	E		VAV.rev.3
VAV.hct.4	Тестирование удобства для пользователя (наблюдения и т.п.)	a	E		
VAV.hct.5	Удобство использования web-приложений, методы тестирования web-сайтов	c	E		
VAV.hct.6	Формальные эксперименты для тестирования гипотез о конкретных элементах управления человеко-машинного интерфейса		D		FND.ef.1
VAV.par	Анализ проблем и создание отчетов			4	
VAV.par1	Анализ отчетов по отказам	c	E		
VAV.par2	Методы отладки и локализации ошибок	a	E		
VAV.par3	Анализ дефектов	k	E		
VAV.par4	Отслеживание проблем	c	E		

4.14. Эволюция программного обеспечения

Описание

Эволюция программного обеспечения является результатом постоянной потребности в поддержке миссии заинтересованных сторон в условиях меняющихся предположений, проблем, требований, архитектур и технологий. Эволюция является неотъемлемой чертой любой работающей программной системы. Поддержка эволюции требует выполнения многих операций как до, так и после выпуска каждой последующей версии или обновлений продуктов, составляющих эволюционирующую систему. Эволюция программного обеспечения – это широкая концепция, расширяющая традиционное понятие сопровождения программного обеспечения.

Модули и темы

Обозначение	Наименование области или модуля знаний	k, c, a	E, D, O	Часы	Связанные темы
EVO.pro	Эволюция программного обеспечения (Software Evolution)			10	
EVO.pro	Эволюционные процессы			6	
EVO.pro.1	Основные концепции эволюции и сопровождения	k	E		
EVO.pro.2	Взаимодействие между эволюционирующими сущностями (предположениями, требованиями, архитектурой, проектом, кодом и т.д.)	k	E		MAA.af.4, DES.ar.4
EVO.pro.3	Модели эволюции программного обеспечения (теории, законы и т.д.)	k	E		
EVO.pro.4	Модели стоимости эволюции		D		FND.ec.3
EVO.pro.5	Планирование эволюции (например, аутсорсинг, собственная разработка и т.д.)		D		MGT.pp
EVO.ac	Эволюционная деятельность			4	VAV.par.4, MGT.cm
EVO.ac.1	Работа с унаследованными системами (legacy systems)(«обертывание» кода (wrappers) и т.д.)	k	E		
EVO.ac.2	Понимание программ (program comprehension) и возвратное проектирование (reverse engineering)	k	E		

EVO.ac.3	Реинжиниринг систем и процессов (с технической точки зрения и с точки зрения бизнеса)	k	E		
EVO.ac.4	Анализ зависимостей (impact analysis)	k	E		
EVO.ac.5	Миграция (с технической точки зрения и с точки зрения бизнеса)	k	E		
EVO.ac.6	Рефакторинг	k	E		
EVO.ac.7	Преобразование программ (program transformation)		D		
EVO.ac.8	Возвратное проектирование данных (data reverse engineering)		D		

4.15. Процессы разработки программного обеспечения

Описание

Процессы разработки программного обеспечения описывают общепринятые модели жизненного цикла программ и устанавливают стандарты процессов. В рамках этой дисциплины изучается определение, реализация, измерение, управление, изменение и улучшение процессов разработки программного обеспечения (ПО), а также использование описанных процессов для осуществления технической и управленческой деятельности, необходимой для разработки и поддержки программного обеспечения.

Модули и темы

Обозначение	**Наименование области или модуля знаний**	**k, c, a**	**E, D, O**	**Часы**	**Связанные темы**
PRO	Процесс разработки программного обеспечения			13	
PRO.con	Основные концепции процессов разработки ПО			3	
PRO.con.1	Тематика и терминология	k	E		
PRO.con.2	Инфраструктура процесса разработки ПО (например, персонал, инструментарий, обучение и т.д.)	k	E		
PRO.con.3	Моделирование и спецификация процессов разработки программного обеспечения	c	E		MAA.rdf.2

PRO.con.4	Измерение и анализ в процессах разработки программного обеспечения	c	E		MGT.ctl.3
PRO.con.5	Улучшение процессов разработки ПО (индивидуальное, командное)	c	E		FND.ef.3, RPO.imp.4,5
PRO.con.6	Анализ и контроль качества (например, профилактика дефектов, процессы инспекций, метрики качества, причинно-следственный анализ)	c	E		MAA.rv.1, VAV.rev, QUA.pda.4
PRO.con.7	Анализ и моделирование моделей процессов разработки программного обеспечения		D		
PRO.imp	Внедрение процессов			10	
PRO.imp.1	Уровни определения процесса (например, организация, проект, команда, индивидуальный)	k	E		
PRO.imp.2	Модели жизненного цикла (гибкие (agile), тяжеловесные, водопадные, спиральные, V-модель и т.д.)	c	E		DES.con.3, VAV.fnd.5
PRO.imp.3	Модели и стандарты процессов жизненного цикла ПО (например, IEEE, ISO и т.д.)	c	E		RPF.pr.5, QUA.pro.2
PRO.imp.4	Индивидуальный процесс (модель, определение, организация, измерение, анализ, улучшение)	c	E		PRO.con.5
PRO.imp.5	Командный процесс (модель, определение, организация, измерение, анализ, улучшения)	c	E		PRO.con.5
PRO.imp.6	Настройка процесса	k	E		
PRO.imp.7	Требования к процессу жизненного цикла ПО (например, стандарт ISO/IEEE 12207)	k	E		PRF.pr.5

4.16. Качество программного обеспечения

Описание

Качество программного обеспечения является пронизывающей концепцией, как затрагивающей все аспекты, так и затрагиваемой всеми аспектами разработки программного обеспечения, поддержкой, инспекциями и сопровождением. Оно охватывает качество разрабатываемых и/или модифицируемых рабочих продуктов (как промежуточных, так и конечных), а также качество рабочих процессов разработки и/или модификации рабочих продуктов. Атрибуты качества продукта включают функциональность, удобство использования, надежность, безопасность, защищенность, сопровождаемость, переносимость, эффективность, производительность и доступность.

Модули и темы

Обозначение	Наименование области или модуля знаний	k, c, a	E, D, O	Часы	Связанные темы
QUA	Качество программного обеспечения			16	
QUA.cc	Концепции и культура качества программного обеспечения			2	
QUA.cc.1	Определение качества	k	E		
QUA.cc.2	Общественная заинтересованность в качестве	k	E		
QUA.cc.3	Стоимость и ущерб от плохого качества	k	E		
QUA.cc.4	Стоимость внедрения моделей качества	c	E		MGT.pp.4
QUA.cc.5	Атрибуты качества программного обеспечения (надежность, удобство использования и т.д.)	k	E		MAA.rva.5, VAV.tst.9, QUA.pda.5
QUA.cc.6	Направления инженерии качества	k	E		
QUA.cc.7	Роли людей, процессов, методов, инструментов и технологий	k	E		
QUA.std	Стандарты качества программного обеспечения			2	PRF.pr.5
QUA.std.1	Системы управления качеством ISO 9000	k	E		
QUA.std.2	Стандарт ISO/IEEE 12207 «Процессы жизненного цикла программного обеспечения»	k	E		
QUA.std.3	Реализация стандартов в организации	k	E		
QUA.std.4	Стандарты IEEE, связанные с качеством		D		
QUA.pro	Процессы управления качеством ПО			4	
QUA.pro.1	Модели и метрики качества программного обеспечения	c	E		VAV.fnd.4, QUA.pda.5
QUA.pro.2	Аспекты моделей процессов разработки программного обеспечения, связанные с качеством	k	E		PRO.imp.3
QUA.pro.3	Введение в ISO 15504 и SEI CMM	k	E		PRF.pr.5
QUA.pro.4	Области процессов ISO 15504, относящиеся к качеству	k	E		PRF.pr.5
QUA.pro.5	Области процессов SW-CMM и CMMI, относящиеся к качеству	k	E		
QUA.pro.6	Критерии премии Болдриджа (Baldrdge award criteria) применительно к программной инженерии		O		
QUA.pro.7	Аспекты качества других моделей процессов		O		

QUA.pca	Обеспечение процесса (process assurance)			4	
QUA.pca.1	Задачи обеспечения процесса	k	E		
QUA.pca.2	Планирование качества	a	E		MGT.pp
QUA.pca.3	Организация и отчетность для обеспечения процесса	a	E		
QUA.pca.4	Методы обеспечения процесса	c	E		
QUA.pda	Обеспечение продукта (product assurance)		E	4	
QUA.pda.1	Сущность обеспечения продукта	k	E		
QUA.pda.2	Отличие обеспечения (assurance) от аттестации и верификации (validation & verification)	k	E		VAV
QUA.pda.3	Модели качества продукта	k	E		
QUA.pda.4	Причинно-следственный анализ и профилактика дефектов	c	E		PRO.con.6
QUA.pda.5	Метрики и измерения качества продукта	c	E		VAV.fnd.4, QUA.cc.5, QUA.pro.1
QUA.pda.6	Оценка атрибутов качества продукта (например, удобство использования, надежность, доступность и т.д.)	c	E		

4.17. Управление программными проектами

Описание

Управление разработкой и сопровождением программного обеспечения использует знания по планированию, организации и мониторингу всех фаз жизненного цикла программного обеспечения. Управление является важным фактором, обеспечивающим следующие моменты: проекты соответствуют организации, скоординирована работа в различных отделах организации, происходит поддержка версий и конфигураций программных продуктов, доступны необходимые ресурсы, работа в проекте разделена надлежащим образом, налажена коммуникация и аккуратно фиксируется продвижение хода работ.

Модули и темы

Обозначение	Наименование области или модуля знаний	k, c, a	E, D, O	Часы	Связанные темы
MGT	Управление программными проектами			19	
MGT.con	Концепции менеджмента			2	
MGT.con.1	Концепции управления проектами	k	E		
MGT.con.2	Классические модели менеджмента	k	E		
MGT.con.3	Роли в управлении проектами	k	E		
MGT.con.4	Структура управления предприятием/организацией	k	E		
MGT.con.5	Методы управления программным обеспечением (например, приобретение, проект, разработка, сопровождение, риски и т.д.)	k	E	EVO	FND.ec.4, MGT.pp.6,
MGT.pp	Планирование проекта			6	VAV.fnd.2, QUA.pca.2
MGT.pp.1	Оценка и планирование	c	E		
MGT.pp.2	Структурная декомпозиция работ	a	E		
MGT.pp.3	Составление календарного плана работ	a	E		
MGT.pp.4	Оценивание трудозатрат	a	E		FND.ec.3, QUA.cc.4
MGT.pp.5	Распределение ресурсов	c	E		
MGT.pp.6	Управление рисками	a	E		FND.ec.4
MGT.per	Организация и управление персоналом			2	PRF.com.3
MGT.per.1	Организационные структуры, должности, ответственности и полномочия	k	E		PRF.psy.1
MGT.per.2	Формальная и неформальная коммуникация	k	E		PRF.com.1, PRF.com.2, PRF.com.3
MGT.per.3	Кадровое обеспечение проекта	k	E		
MGT.per.4	Обучение персонала, карьерное развитие, оценка	k	E		
MGT.per.5	Управление совещаниями	a	E		
MGT.per.6	Формирование и мотивация команды	a	E		
MGT.per.7	Разрешение конфликтов	a	E		

MGT.ctl	Контроль проекта			4	
MGT.ctl.1	Контроль изменений	k	E		MAA.rdf.5, MGT.cm.1,2
MGT.ctl.2	Мониторинг и отчетность	c	E		
MGT.ctl.3	Измерение и анализ результатов	c	E		PRO.con.4
MGT.ctl.4	Корректирование и восстановление	k	E		
MGT.ctl.5	Поощрения и взыскания		O		
MGT.ctl.6	Стандарты производительности		O		
MGT.cm	Управление конфигурацией программного обеспечения			5	
MGT.cm.1	Контроль версий	a	E		MGT.ctl.1
MGT.cm.2	Управление выпуском версий	c	E		MGT.ctl.1
MGT.cm.3	Инструментальная поддержка	c	E		
MGT.cm.4	Сборки	c	E		
MGT.cm.5	Процесс управления конфигурацией программного обеспечения	k	E		
MGT.cm.6	Проблемы сопровождения ПО	k	E		EVO.ac
MGT.cm.7	Распространение и резервное копирование		D		

4.18. Системные и прикладные специальности

Частью процесса обучения студентов является специализация в одной или нескольких областях. В пределах своей специальности студентам следует изучать материал гораздо глубже предложенного выше базового материала. Они могут специализироваться в одной из десяти или более вышеизложенных областях знаний либо же могут специализироваться по одной или нескольким нижеприведенным прикладным областям. Для каждой прикладной области студенты должны получить представление о родственных областях знаний параллельно с глубокими знаниями о проектировании определенных систем. Студенты также должны узнать о характеристиках типичных продуктов этих областей и как эти характеристики влияют на проектирование и разработку системы. Каждая прикладная специальность, приведенная ниже, дополнена списком родственных тем, которые требуются для поддержки приложения.

Список прикладных областей не задумывался исчерпывающим, а скорее создавался как руководство по разработке учебных планов этих специальностей

Специальности и связанные с ними темы

SAS	Системные и прикладные специальности
SAS.net	Распределенные системы
SAS.net.1	Знания и навыки в области web-технологий
SAS.net.2	Углубленное изучение сетей
SAS.net.3	Углубленное изучение защиты информации
SAS.inf	Информационные системы и обработка данных
SAS.inf.1	Углубленное изучение баз данных
SAS.inf.2	Углубленное изучение бизнес-управления
SAS.inf.3	Хранилища данных
SAS.fin	Финансовые системы и системы электронной коммерции
SAS.fin.1	Бухгалтерский учет
SAS.fin.2	Финансы
SAS.fin.3	Углубленное изучение защиты информации
SAS.sur	Отказоустойчивые и живучие (survivable) системы
SAS.sur.1	Знания и навыки для гетерогенных распределенных систем
SAS.sur.2	Углубленное изучение защиты информации
SAS.sur.3	Анализ отказов и восстановление
SAS.sur.4	Выявление несанкционированного вмешательства
SAS.sec	Хорошо защищенные системы
SAS.sec.1	Вопросы бизнеса, связанные с защитой информации
SAS.sec.2	Информационная уязвимость и риски
SAS.sec.3	Криптография, криптоанализ, стеганография и т.п.
SAS.sec.4	Углубленное изучение компьютерных сетей
SAS.sfy	Системы с повышенными требованиями к безопасности
SAS.sfy.1	Углубленное изучение формальных методов, доказательство правильности и т.п.
SAS.sfy.2	Знания по контролю над системами
SAS.sfy.3	Углубленное изучение моделей отказов, анализ эффектов и дерево отказов
SAS.emb	Встроенные системы и системы реального времени
SAS.emb.1	Аппаратное обеспечение для встроенных систем
SAS.emb.2	Языки и средства разработки
SAS.emb.3	Углубленное изучение вопросов синхронизации (timing issues)
SAS.emb.4	Верификация аппаратного обеспечения

SAS.bio	Биомедицинские системы
SAS.bio.1	Биология и родственные науки
SAS.bio.2	Связанные знания по системам, критичным к ошибкам
SAS.sci	Научно-исследовательские системы
SAS.sci.1	Углубленное изучение связанных наук
SAS.sci.2	Углубленное изучение статистики
SAS.sci.3	Визуализация и графика
SAS.tel	Телекоммуникационные системы
SAS.tel.1	Углубленное изучение теории сигналов, теории информации и т.п.
SAS.tel.2	Телефония и протоколы телекоммуникации
SAS.av	Авиационное электронное оборудование и транспортные системы
SAS.av.1	Концепции машиностроения
SAS.av.2	Смежные знания по системам с повышенными требованиями к безопасности
SAS.av.3	Смежные знания по встроенным системам и системам реального времени
SAS.ind	Системы контроля промышленного процесса
SAS.ind.1	Системы контроля
SAS.ind.2	Производственная инженерия и другие важные инженерные области
SAS.ind.3	Смежные знания по встроенным системам и системам реального времени
SAS.mm	Мультимедийные, игровые и развлекательные системы
SAS.mm.1	Визуализация, осязание (haptics) и графика
SAS.mm.2	Углубленное изучение проектирования человеко-машинного интерфейса
SAS.mm.3	Углубленное изучение компьютерных сетей
SAS.mob	Системы для малых и мобильных платформ
SAS.mob.1	Беспроводные технологии
SAS.mob.2	Углубленное изучение человеко-машинного интерфейса для малых и мобильных платформ
SAS.mob.3	Смежные знания по встроенным системам и системам реального времени
SAS.mob.4	Смежные знания по телекоммуникационным системам
SAS.ab	Системы, основанные на агентах (agent-based systems)
SAS.ab.1	Машинное обучение
SAS.ab.2	Нечеткая логика
SAS.ab.3	Инженерия знаний

ГЛАВА 5. Рекомендации по разработке учебных планов и преподаванию программной инженерии

В главе 4 мы представили преподаваемый материал по программной инженерии (SEEK), которым должны владеть все выпускники, специализирующиеся в программной инженерии. Однако «как преподавать» материал по программной инженерии – вопрос не менее важный, чем «что преподавать». В данной главе приведены рекомендации, адресованные как разработчикам учебных планов по программной инженерии, так и преподавателям отдельных курсов.

5.1. Рекомендации для разработчиков учебных планов и преподавателей

Рекомендация 1. Разработчики учебных планов и преподаватели должны иметь достаточные знания и опыт в соответствующих областях, а также понимать характерные свойства программной инженерии.

Разработчики учебных планов и преподаватели должны иметь широкую эрудицию в области программной инженерии, что подразумевает:

- наличие знаний по программной инженерии в большинстве областей SEEK;
- практический опыт работы в программной инженерии;
- общественное признание собственных знаний в сфере программной инженерии, полученное либо путем публикации работ по данной тематике, либо активным участием в соответствующих профессиональных сообществах;
- постоянное ознакомление с новыми предметными областями (application domains) программной инженерии (например, использование программной инженерии в других инженерных дисциплинах или в сфере бизнеса), при этом не претендуя на звание эксперта в данных областях;
- наличие мотивации и возможностей для поддержания высокой осведомленности о последних разработках в данной дисциплине.

Несоблюдение перечисленных выше принципов подвергнет учебную программу или курс следующим рискам:

- Программа или курс могут быть излишне перегружены информацией о каком-то одном типе программного обеспечения или классе методов, не обеспечивая тем самым достаточно широкого освещения изучаемой области или даже формируя у студентов искаженное представление о предмете. Например, преподаватели, работавшие только с системами реального времени или системами обработки данных, подвержены риску перегрузки своих курсов информацией об изученных ими системах. В том, что некоторые учебные курсы специализируются на каких-то конкретных направлениях программной инженерии, нет ничего плохого, однако

подобные специализации должны быть явно отражены в названиях курсов. Кроме того, в рамках учебной программы в целом студенты должны получить знания о максимально широком спектре систем и подходов.

- Профессорско-преподавательский состав, составленный из специалистов по теоретической информатике, может плохо справиться с донесением до студентов важности инженерных аспектов программной инженерии.
- Преподаватели из родственных инженерных дисциплин подвержены опасности чтения курсов по программной инженерии без должного понимания основ информатики, являющихся основой для большей части программирования. Кроме того, не все преподаватели с таким опытом могут продемонстрировать применение программного обеспечения в достаточно широком спектре прикладных областей, помимо собственно инженерных приложений.
- Преподаватели, не имеющие опыта участия в разработке больших систем, могут недостаточно ценить важность процессов, качества, эволюции и менеджмента, являющихся областями знаний SEEK.
- Преподаватели, сделавшие исследовательскую карьеру в экспериментальных и новых методах программной инженерии, не всегда понимают, что студентов необходимо прежде всего обучать тому, что они смогут сразу применить на практике, а также пониманию практических и теоретических обоснований преподаваемого материала.

5.2. Рекомендации по разработке учебного плана

Рекомендация 2. Разработчики учебных планов и преподаватели должны мыслить в терминах конечного результата.

При разработке программы обучения в целом и отдельных курсов в частности необходимо уделять внимание результатам и целям обучения. Более того, результаты обучения должны учитываться и при преподавании отдельных курсов. Нацеленность на конечный результат поможет преподавателям включать в учебные планы только релевантные материалы, а также обеспечить его преподавание надлежащим образом и на соответствующем уровне детализации.

Результаты обучения студентов по программе SE2004 (см. главу 2) должны использоваться в качестве основы при разработке и оценке общих учебных планов по программной инженерии. Они также могут быть конкретизированы для отдельных курсов.

Кроме того, некоторые учебные заведения могут определять дополнительные цели обучения (например, навыки в специальных прикладных областях или углубленные знания в некоторых областях знаний SEEK).

Рекомендация 3. Разработчики учебных планов должны поддерживать баланс между полнотой покрытия материала и гибкостью построения курсов, оставляя преподавателям простор для инноваций.

Среди разработчиков учебных планов существует тенденция включать в программу или курс большой перечень информации, которую «крайне необходимо» раскрыть при обучении, что не оставляет преподавателям никакой свободы при изложении курсов и не предоставляет возможностей для более глубокого изложения (пусть даже в ущерб широте освещения).

Однако существует также распространенное мнение, что студент, получивший фундаментальные знания, а также имеющий общее представление о дополнительных темах, сможет заполнить пробелы в своем обучении позже – возможно, уже в процессе профессиональной деятельности и по мере необходимости. Из этого следует, что некоторые углубленные темы, посвященные процессам разработки ПО и отмеченные в SEEK как материалы уровня «a» (уровень применения), могут быть изучены на уровне «k» (уровень знания), если это абсолютно необходимо для инноваций в преподавании. Однако в целом не следует допускать снижения уровня изучения углубленного технического или математического материала с уровня «a» до уровня «k», так как изучение данного материала на рабочем месте будет сильно затруднено.

Рекомендация 4. Многие ключевые концепции, принципы и проблемы программной инженерии нужно преподавать как сквозные темы в течение всего учебного плана для формирования у студентов инженерного склада ума.

Некоторые темы SEEK рекомендуется равномерно распределять по различным курсам учебного плана. При таком подходе ранние курсы служат введением в тему, а последующие курсы закрепляют и расширяют полученные студентами знания. В большинстве случаев необходимо также предусмотреть курсы или части курсов, обеспечивающие углубленное изучение данной темы.

Помимо материалов по этике и использованию инструментальных средств, о которых мы будем говорить подробнее в последующих рекомендациях, следующие темы рекомендуется преподавать как минимум частично, в виде повторяющихся сквозных тем:

- Измерение, количественный анализ, формальные и математические методы.
- Моделирование, представление и абстракция.
- Человеческий фактор и удобство использования – студентам необходимо постоянно напоминать, что программная инженерия не сводится к технологии.
- Наблюдение, что многие принципы программной инженерии являются, по сути, базовыми инженерными принципами. Студенты смогут лучше понять принципы программной инженерии, если увидят примеры применимости этих же принципов в других дисциплинах, например тот факт, что все инженеры широко используют модели, измерения, решают проблемы, используют «черные ящики» и т.д.
- Важность масштаба – студенты могут практиковаться на решении только относительно небольших проблем, однако должны понимать, что

мощь некоторых методов наиболее полно проявляется в больших системах. Студенты должны мысленно соотносить свои практические задания с масштабом промышленных систем, а также учиться читать и понимать код больших систем.

- Важность повторного использования (reuse).
- Большая часть материала по таким областям знаний, как процессы, качество, эволюция и менеджмент.

Рекомендация 5. Изучение некоторых тем программной инженерии требует определенного уровня зрелости, поэтому они должны преподаваться ближе к концу учебного плана. С другой стороны, необходимо предусмотреть преподавание материалов, способствующих формированию зрелости у студентов, на более ранних стадиях.

Учебный материал должен быть структурирован таким образом, чтобы студенты полностью оценили важность и мотивацию основных принципов. Раннее изучение некоторых тем SEEK, например связанных с процессами, качеством, эволюцией и менеджментом, скорее всего, приведет к слабому пониманию и низкой оценке важности предмета студентами. Необходимо принимать это во внимание при разработке последовательности преподавания материала и методов ознакомления студентов с реальным практическим опытом. Рекомендуется прочитать вводные курсы как можно раньше в процессе обучения, но в то же время отложить преподавание основной части материала как можно позже.

С другой стороны, практические занятия необходимы студентам и на ранних стадиях обучения, чтобы они могли овладевать предметом, решая определенные практические задачи (в рабочих группах либо студенческих проектах). Примерами тем для раннего обучения являются программирование, человеческие факторы, аспекты формирования требований и проектирования, а также верификация и аттестация. Это не означает, что программирование должно изучаться в первую очередь (как и в традиционном курсе CS1), однако значительная часть материала по программированию должна быть изучена в течение первого года обучения.

На начальной стадии обучения студентов необходимо также знакомить со «сложными» задачами программной инженерии. Примерами таких задач могут служить задачи, связанные с изменяющимися требованиями, задачи понимания и изменения существующих больших систем, работа в большой группе и др. Основная идея проведения таких занятий – развить у студентов понимание значимости таких областей знаний, как процесс, качество, эволюция и менеджмент до начала изучения этих областей.

Рекомендация 6. Студенты должны также изучить какую-либо предметную область (или области), не относящуюся к программной инженерии.

Наибольшая часть деятельности в области программной инженерии связана с поиском решений для задач в конкретных предметных областях, не относящихся к программной инженерии. Поэтому в учебном плане обходимо предусмотреть изучение студентами в разумных объемах одной или нескольких «внешних» предметных областей.

Изучение данного материала не только даст студентам знания, которые они смогут применять к решению задач программной инженерии, но также научит их терминологии и процессам, присущим данной предметной области, подготавливая почву для дальнейшего более углубленного изучения.

Под «разумными объемами» понимается один или два курса, имеющих уровень выше вводного (углубленные курсы на втором и выше годах обучения). Выбор предметной области (областей) остается за разработчиками плана, а во многих случаях может быть частично оставлен на усмотрение студентов. Предметные области могут включать другие направления инженерии или естественных наук, бизнес-приложения, социальные или гуманитарные науки. При этом ни одна предметная область не должна рассматриваться как «более важная», чем другие.

Изучение определенных предметных областей может потребовать включения дополнительных курсов, таких как некоторые области математики и информатики или же углубленные области программной инженерии. Отсылаем заинтересованных читателей к разделу «Системные и прикладные специальности» в конце SEEK (глава 4), в котором приведены рекомендации для подобных вспомогательных курсов.

Данная рекомендация не исключает проектирования курсов или программ, в которых тесно интегрировано обучение знаниям предметной области с обучением программной инженерии. Напротив, такой подход является инновационным и достойным похвалы. Учебное заведение, например, может разработать курс под названием «Программная инженерия для телекоммуникаций», «Программная инженерия для аэрокосмической промышленности», «Программная инженерия для информационных систем» или «Программная инженерия для звуковых и музыкальных систем». Однако в таких случаях особое внимание необходимо уделить балансу глубины изучения конкретной предметной области и программной инженерии. Существует риск того, что преподаватель, его лекции и учебные материалы могут не дать студентам достаточного уровня знаний в одной из этих двух областей.

5.3. Характеристики учебных планов и процесса преподавания

Рекомендация 7. Курсы по программной инженерии должны демонстрировать принадлежность этой дисциплины как к компьютингу, так и к инженерии.

Преподаватели должны продемонстрировать студентам аспекты программной инженерии, которые она разделяет как с другими инженерными дисциплинами, так и с другими дисциплинами компьютинга, в особенности информатики. Характерные свойства инженерии и компьютинга представлены в главе 2.

- **Инженерия**. Инженерные дисциплины развивались тысячелетиями и приобрели за это время большой объем общих знаний. Тем не менее определенные аспекты инженерии требуют адаптации для программной инженерии. Студенты, изучающие программную инженерию, должны осознать, что они являются настоящими инженерами: они должны выработать в себе инже-

нерный дух, а также понимание степени ответственности, накладываемой на инженеров. Этого можно достичь соответствующим отношением со стороны профессорско-преподавательского состава и администрации.

Этот принцип не требует от программных инженеров одобрения всех аспектов инженерии без исключения. Некоторые аспекты этой профессии критикуются как ее представителями, так и представителями других специальностей, и эту критику нельзя не учитывать при развитии профессии. Существует также несколько свойств, отличающих программную инженерию от других типов инженерии (например, производство нематериальной продукции и глубокие связи с другими разделами науки), которые также необходимо учитывать. Этот принцип не требует также и принятия какой-либо одной конкретной модели профессии.

- **Компьютинг**. Для обеспечения профессиональной компетентности, позволяющей разрабатывать программное обеспечение высокого качества, программные инженеры должны иметь твердые и глубокие базовые знания в области информатики (см. главу 4). Эти знания помогут им понять границы применимости компьютинга, а также определить технологии, подходящие для данного программного проекта.

  Этот принцип не требует, чтобы знания программного инженера в данной области были столь же глубоки, как и у специалистов в области информатики. Однако эти знания и опыт должны быть достаточны, чтобы сделать правильный выбор среди существующих технологий и правильно их применять. Программный инженер должен также иметь достаточное понимание сложности и границ применимости данных технологий, чтобы при необходимости принять решение о консультации с соответствующими специалистами (например, специалистами по базам данных).

Рекомендация 8. Студенты должны получить определенные навыки, выходящие за рамки данного предмета.

Перечисленные ниже навыки требуются практически во всех сферах деятельности, с которыми могут столкнуться студенты после выпуска из университета. Эти навыки приобретаются преимущественно на практике.

- **Критическое мышление.** Способность критически оценить конкурирующие решения – одна из основных черт, присущих профессиональному инженеру. Поэтому учебный план и методы преподавания должны помочь студентам приобрести необходимые знания, навыки анализа и методы эффективной оценки. Важной чертой является стремление к критическому мышлению. Студентов необходимо также обучить умению оценивать надежность различных источников информации.
- **Оценка и оспаривание получаемых знаний.** Студенты должны научиться не принимать на веру любую информацию от преподавателей или из книг. Они должны также понимать ограниченность современных знаний по программной инженерии и направления, в которых эти знания должны развиваться.

- **Осознание собственных ограничений.** Студентам необходимо объяснить, что профессионалы зачастую прибегают к консультированию у других профессионалов, а также показать преимущества командной работы.
- **Эффективная коммуникация.** Студенты должны научиться эффективно обмениваться информацией различными способами: письменно, при проведении презентаций, демонстраций собственных и чужих программных разработок, а также во время различного рода обсуждений. Студенты должны развить навыки восприятия на слух, навыки сотрудничества и переговоров.
- **Этичное и профессиональное поведение.** Студенты должны научиться контролировать соответствие своей деятельности нормам этики, конфиденциальности (privacy) и безопасности. Дополнительная информация на эту тему содержится в рекомендации 15.

Некоторые из перечисленных выше тем (особенно навыки коммуникации) можно освещать в лекциях. Однако студенты смогут усвоить данные навыки более эффективно при их постоянном использовании в групповых проектах, письменных работах и студенческих презентациях.

Рекомендация 9. Студентам необходимо привить стремление учиться и саморазвиваться.

Так как большая часть изучаемого изменится в течение будущей профессиональной деятельности студентов, и лишь малая часть необходимой для изучения информации преподается в университете, крайне важно, чтобы студенты развивали в себе навыки постоянного расширения собственных знаний.

Рекомендация 10. Программная инженерия должна преподаваться как дисциплина, ориентированная на решение проблем.

Важнейшей задачей проектов по разработке программного обеспечения является разрешение проблем заказчика – как явных, так и неявных. Необходимо учитывать это при составлении программ и курсов. Осознание данного факта поможет студентам рационально подходить к процессу обучения, облегчит понимание материала и поможет оценить соответствие материала целям курса. К сожалению, наиболее частой ошибкой является фокусирование исключительно на технических проблемах, что в дальнейшем приводит к созданию непригодных для использования систем.

Существует множество классов важных проблем. Некоторые из них, такие как проблемы анализа, проектирования и тестирования, относятся к продукту и напрямую связаны с разрешением проблем заказчика. Другие же, такие как улучшение процесса, являются мета-проблемами, решение которых ускорит процесс разрешения проблем, относящихся к продукту и к разрешению проблем заказчика. Кроме того, существует ряд проблем, не входящих в эти две категории (например, этические проблемы).

Навыки решения проблем наилучшим образом развиваются при анализе примеров и решении учебных задач во время практических занятий. Демонстра-

ция решения на экране преподавателем является необходимым, но недостаточным условием развития необходимых навыков. Поэтому студентам необходимо давать значительное количество практических задач для самостоятельного решения.

Рекомендация 11. Наибольший упор в обучении необходимо делать на основополагающие и неизменные *принципы* программной инженерии, а не на информацию о *новейших или специализированных средствах* и инструментах.

SEEK содержит большое количество тем, для изучения которых необходимо использовать различное аппаратное обеспечение, программные приложения, технологии и процессы (которые мы в совокупности будем называть средствами и инструментами). В хорошем учебном плане наибольшее внимание необходимо уделять разделам SEEK, не меняющимся с течением времени, а не изучению деталей каких-либо инструментов. Данные темы должны сохранять свою значимость в течение многих лет; опыт и знания, полученные от изучения данных тем, должны быть применимы как через 10, так и через 20 лет. Отдельные же программные средства и инструменты будут часто меняться. К примеру, будет ошибочным чрезмерное фокусирование на конкретном программном обеспечении либо углубленное изучение стадий какой-либо методологии или синтаксиса одного из языков программирования.

Применяя эту рекомендацию к обучению языкам программирования, необходимо отметить, что граница между постоянными и меняющимися знаниями трудно определима и может меняться. Нет сомнений, что программные инженеры должны детально изучить несколько языков программирования, а также языки других типов (например, языки спецификаций). Данную рекомендацию следует интерпретировать так, что при изучении этих языков студенты должны освоить гораздо больше, чем поверхностный синтаксис, чтобы последующее ознакомление с новыми языками требовало минимальных усилий.

Применение данной рекомендации к процессам (известным также как «методы» или «методологии») сходно с ее применением к языкам. Студентам следует запоминать не длинные перечни шагов, а понимать глобальные идеи, стоящие за этими перечнями, чтобы при появлении новых методологий студенты могли бы творчески подойти к проблеме их адаптации и использования различных процессов.

Для технологий (как аппаратных, так и программных) данная рекомендация означает, что нет необходимости детально запоминать программный интерфейс приложений (API), систему команд конкретной ЭВМ или пользовательский интерфейс, просто чтобы помнить их наизусть. Вместо этого студенты должны развить в себе навыки поиска необходимой информации в справочной литературе и руководствах, концентрируясь при этом на более важных задачах.

Рекомендация 12. Обучение должно проходить с использованием соответствующих современных средств, даже если данные средства не являются целью обучения.

Программная инженерия требует осуществлять обоснованный выбор и эффективное использование соответствующего аппаратного обеспечения, программных средств, технологий и процессов (как и раньше в совокупности назы-

ваемых средствами). Поэтому для студентов должен быть привычным процесс выбора различных средств, и с этой привычкой они должны начать свою профессиональную деятельность – привычкой, которую трудно приобрести на рабочем месте, когда необходимость скорейшего предоставления результатов часто затрудняет изучение новых средств.

Необходимо тщательно выбирать наиболее пригодные для обучения средства. Слишком сложное, ненадежное, дорогое, трудно осваиваемое в необходимый срок и в данной аудитории, на данных ресурсах или дающее очень мало преимуществ от изучения средство, будет непригодным как для обучения, так и для использования в профессиональной деятельности. Множество средств программной инженерии оказались неудачными, так как они не соответствовали указанным критериям.

Необходимо выбирать такие средства, которые бы поддерживали принципы обучения.

В учебном процессе необходимо использовать самые современные средства и инструменты, и на то есть несколько причин:

a) чтобы студенты могли использовать данные средства на своем будущем рабочем месте, осуществляя в некотором роде распространение технологий;
b) чтобы студенты могли воспользоваться преимуществами от приобретения навыков работы с изученными средствами;
c) чтобы студенты и работодатели не воспринимали обучение в целом как устаревшее – вне зависимости от использования в нем новейших концепций. Следует отметить, однако, что более ранние средства и инструменты иногда могут быть проще и поэтому лучше подходить для решения определенных учебных задач.

Может показаться, что данная рекомендация противоречит рекомендации 11, однако это не так. Чтобы исключить противоречие, необходимо понять, что обучение использованию средств не означает, что целью обучения являются сами средства. Обучение использованию средств должно быть второстепенной задачей, решаемой в лабораторных условиях при работе с консультантом (тьютором) либо при самостоятельном изучении. Студенты должны понимать, что средства лишь помогают решать основные задачи, и они должны учиться не бояться знакомства с новыми средствами.

Рекомендация 13. Материал, преподаваемый в рамках учебного плана по программной инженерии, должен быть связан с серьезными исследованиями и иметь строгое математическое или иное теоретическое обоснование или же являться общепринятой профессиональной практикой.

Необходимо иметь подтверждение того, что преподаваемые знания соответствуют действительности и применимы. Таким подтверждением может быть проверенная математическая теория (как это происходит во многих областях информатики) или же широко используемые и общепринятые практические принципы.

Важно, однако, не быть излишне догматичным в отношении теории: это не всегда приемлемо. Например, формализация спецификации или дизайна с при-

менением математических подходов может быть неэффективна и громоздка. Однако в некоторых обстоятельствах такой подход может быть необходим.

В ситуациях, когда преподаваемый материал основывается на преимущественно общепринятых принципах, не подвергавшихся ранее научной проверке, необходимо четко оговаривать, что материал не является бесспорным.

Обучение зарекомендовавшим себя подходам (best practice) не должно быть контекстно-независимым, напротив – необходимо приводить примеры успешного использования указанных советов и проблем, возникших в результате их игнорирования. Этот принцип относится и к представлению данных, полученных в результате исследований.

Данная рекомендация дополняет рекомендацию 11. В то время как рекомендация 11 говорит о необходимости обучения фундаментальным принципам программной инженерии, в рекомендации 13 говорится о необходимости тщательного обоснования преподаваемого материала.

Рекомендация 14. Учебный план должен основываться на реальном практическом опыте.

Включение элементов реального окружения в учебный план необходимо для эффективного развития навыков работы в области программной инженерии, а также для изучения концепций данной области знаний. Программа должна быть построена таким образом, чтобы как минимум включать следующее:

- **Учебные примеры (case studies).** Изучение существующих систем, а также учебных примеров (case studies) проектов, с целью их критической оценки и повторного использования наиболее успешных.
- **Проектно-ориентированные курсы (project-based classes).** Некоторые курсы должны воспроизводить типичные проекты данного направления профессиональной деятельности. Занятия должны включать работу в группах, презентации, формальное рецензирование (review), обеспечение качества и др. Дополнительную пользу принесет включение в курс обучения данных об одном из существующих заказчиков или ряде заказчиков. Групповые проекты могут быть междисциплинарными. Студенты должны получить опыт работы в различных ролях, присущих группе программных инженеров: в качестве менеджера проекта, разработчика инструментальных средств, разработчика требований и др.
- **Дипломный проект.** Студентам необходимо дать возможность работы над крупным проектом, предпочтительно занимающим весь последний год обучения и позволяющим применить на практике полученные знания и навыки. В отличие от проектно-ориентированных курсов в процессе работы над дипломным проектом все решения принимаются самими студентами для решения выбранной ими задачи. Информация по дипломному проекту содержится в разделе 6.3.2. В одних учебных заведениях приняты групповые дипломные проекты, в то время как в других – индивидуальные.

- **Практические задания.** Студентам необходимо выдавать практические упражнения, помогающие развить навыки работы с современными процедурами и процессами.
- **Опыт работы студентов.** По возможности, в программу необходимо включить прохождение студентами практики на предприятиях данной отрасли. Это может быть организовано в форме интернатуры (одной или более), совмещения обучения и работы, либо же посеместровое чередование обучения и работы по специальности (sandwich work). При существовании такой возможности желательно также сделать производственную практику обязательным элементом обучения. В случае если организация производственной практики затруднена, необходимо имитировать рабочие условия при чтении курсов.

Несмотря на описанное выше, преподаватели должны помнить, что производственный опыт, который могут получить студенты во время обучения, ограничен: студенты смогут оценить сложность работы и последствия плохой работы, только получив реальные результаты своей деятельности в различных проектах в течение профессиональной карьеры. Преподаватели могут оказать лишь начальную помощь студентам в постижении реалий жизни и должны осознавать, что обучение студентов пониманию истинного смысла получаемых знаний является сложной задачей.

Рекомендация 15. Необходимо как можно чаще поднимать вопросы, связанные с моральными, юридическими и экономическими аспектами будущей деятельности студентов, а также раскрывать понятие того, что значит быть профессионалом в выбранной ими области.

Одной из основных задач каждой профессии является обеспечение соблюдения этических и профессиональных норм ее представителями. При обсуждении этих принципов на протяжении всего обучения они глубоко укоренятся в сознании студентов. Одним из возможных методов достижения этого является знакомство студентов с соответствующими стандартами и рекомендациями. В разделе 3.3 содержится более подробная информация на тему профессионализма.

5.4. Общие стратегии педагогики в области программной инженерии

Рекомендация 16. Чтобы обеспечить понимание студентами определенных важных идей, необходимо мотивировать их интересными, конкретными и убедительными примерами.

В некоторых случаях к необходимости изучения наиболее важных в данной дисциплине концепций и методов будущие программные инженеры могут придти вследствие получения горького опыта. Иногда преподавательский состав не осознает значения данных концепций и поэтому не обучает им студентов. В других случаях преподаватели сталкиваются со скептическим отношением к данной теме со стороны части студентов.

Во всех этих ситуациях необходимо уделить особое внимание мотивации студентов к восприятию необходимых концепций, приводя познавательные, конкретные и соответствующие данной теме примеры. Примеры должны быть достаточного объема и сложности, чтобы показать, что применение изучаемых принципов на практике несет в себе очевидные преимущества, и нежелание использовать указанные принципы приводит к отрицательным результатам.

Ниже приводятся примеры областей, изучение которых особо нуждается в мотивировании:

- *Основы математики*. Логика и дискретная математика должны преподаваться в контексте их применения к решению задач программной инженерии или информатики. Если обучение включает в себя выводы и доказательства, оно должно объяснять важность полученных выводов. Статистические и эмпирические методы должны также преподаваться в свете возможностей их применения.
- *Процесс и качество*. Студентам необходимо раскрывать последствия плохо организованного процесса и низкого качества работы. Необходимо также помочь студентам формировать хорошие процессы и обеспечивать качественные результаты, чтобы они на себе ощутили эффект улучшения, гордость за проделанную работу и учились ценить ее качество.
- *Человеческий фактор и удобство использования программного обеспечения*. Студенты чаще всего не считают необходимым уделять внимание данным факторам, пока не ощутят на себе проблемы использования программного обеспечения или не увидят, какие трудности испытывают пользователи программного обеспечения.

Рекомендация 17. В XXI веке обучение программной инженерии должно выйти за пределы лекционного формата, поэтому необходимо обратить внимание на важность обучения различным подходам к преподаванию и получению знаний.

Наиболее распространенный подход к преподаванию программной инженерии – лекционный, дополняемый лабораторными работами, индивидуальной работой с преподавателем и др. При этом использование альтернативных подходов может способствовать более эффективному обучению студентов. Некоторые подходы могут быть приняты как дополнение или даже замена большей части обучения в лекционном формате. Они включают:

- *Проблемно-ориентированное обучение*. Этот подход доказал свою полезность в других профессиональных областях и сейчас применяется некоторыми учреждениями для обучения программной инженерии. В рекомендации 10 содержится дополнительная информация об ориентированности программной инженерии на поиск решений проблем.
- *Обучение «точно в срок» (just-in-time learning)*. Например, обучение фундаментальным основам предмета непосредственно перед обучением возможностям применения полученных знаний или обучение математическим методам за день до обучения их применению в контексте про-

граммной инженерии. Существуют свидетельства того, что подобный подход помогает студентам усвоить основы материала, однако он трудно реализуем, так как вызывает проблемы координации учебных курсов.

- *Обучение на трудностях (learning by failure)*. Студентам дают трудно разрешимую задачу. Затем обучают методам, позволяющим в будущем легко решать задачи подобного типа.
- *Материалы для самообучения, прорабатываемые студентами самостоятельно*. Сюда входят online-обучение и обучение с использованием компьютеров.

Рекомендация 18. Необходимая эффективность обучения и обеспечение синергетического эффекта могут быть достигнуты при проектировании учебного плана таким образом, чтобы студенты получали несколько типов знаний одновременно.

Множество людей, просматривая SEEK, заметили, что он содержит большое количество материала и, наоборот, на изучение многих тем выделено небольшое количество часов. Однако, если правильно построить учебный план, большое количество тем может изучаться одновременно; и в действительности две темы, которым выделено x и y часов соответственно, могут изучаться совместно за время, меньшее, чем x+y.

Ниже перечислены некоторые из множества ситуаций, в которых возможно применение метода совместного обучения:

- *Моделирование, языки и нотации*. Глубина понимания особенностей таких языков, как UML достигается путем его использования при изучении других концепций. То же самое касается формальных методов и программирования. Необходимо отдельно выделить некоторое время для изучения основ языков и подходов моделирования, но дальнейшее расширение и углубление знаний студентами может достигаться при изучении множества других тем.
- *Процесс, качество и менеджмент*. Студентов можно обучить следованию определенным процессам при работе над задачами и проектами, основная цель которых – обучение другим концепциям. При этом желательно ознакомить студентов с процессом, чтобы они понимали, почему им предлагают использовать тот, а не иной процесс. При этом также желательно проследить решение данной задачи или реализацию проекта, обсуждая одновременно полезность применения данного процесса. С большой долей вероятности подобный подход поможет вам добиться значительной глубины понимания материала студентами, при этом время, которое подобное обучение отнимет у основного материала, будет довольно незначительным.
- ***Математика.*** Студенты могут углублять и расширять свои познания в статистике, проводя анализ данных, полученных при изучении надежности или производительности систем. Можно также найти массу возможностей углубленного изучения логики или других направлений дискретной математики.

- **Одновременное обучение нескольким темам** поможет студентам установить взаимосвязи между предметами и может повысить их интерес к изучению материала. В любом случае это ведет к лучшему усвоению материала.

Рекомендация 19. Учебный план и входящие в него курсы должны регулярно пересматриваться и обновляться.

Программная инженерия как наука быстро развивается; вследствие этого большая часть курсов (если не все), да и сам учебный план могут оказаться устаревшими. Учебные заведения и преподаватели должны регулярно просматривать свои курсы и программы и изменять их при необходимости. Эта рекомендация относится к учебному плану или курсам, самостоятельно разработанными вузами или же преподавателями. С другой стороны, принципы 3 и 4 в разделе 2.1 требуют и от SE2004 признания быстрого развития данного направления и внесения всех необходимых изменений в публикуемые материалы.

5.5. Заключительные замечания

Приведенная выше информация представляет собой набор основных рекомендаций, закладывающих основу разработки высококачественной программы по обучению программной инженерии. Разумеется, они не охватывают все потенциально возможные проблемы. Каждое учебное заведение, скорее всего, будет иметь местные либо национальные потребности, обусловливаемые производственной спецификой, государственной политикой и др. Необходимо также принимать во внимание стремления самих студентов. Студенты должны понимать важность образования и должны видеть, как оно соответствует их потребностям. Чаще всего это обусловливается их возможностями (к примеру, разработанными ими системами) во время обучения, видением своей карьеры и возможностями выбора, стоящими перед студентами. Безусловно, они должны развить в себе уверенность в способности выдержать конкуренцию на международном рынке труда.

Любой учебный план по программной инженерии должен объединить все эти рекомендации в единую, логически последовательную программу. В идеале отдельные курсы и среда, в которой производится обучение, должны иметь единый стиль. Процесс обучения в области программной инженерии должен привить студентам ожидания и ценности, требуемые при создании высококачественных программных систем.

ГЛАВА 6. Курсы и порядок их преподавания

В данной главе приводится ряд примеров учебного плана, которые могут быть использованы для преподавания SEEK согласно рекомендациям, сформулированным в главе 5.

Данная глава имеет следующую структуру. В разделе 6.1 приведено описание категорий, на которые разделены курсы, а также используемой схемы кодирования. В последующих разделах приведены шаблоны вводных курсов, промежуточных курсов по программной инженерии и других курсов. Детали курсов, включая взаимосвязь с SEEK, описываются в приложении А.

Данный документ создавался в качестве ресурса для образовательных учреждений, в которых разрабатывают или совершенствуют учебные планы в области программной инженерии для преподавания в вузах, а также для аккредитационных агентств, которым необходим пример учебного плана для оценки учебных программ различных учебных заведений. Шаблоны и описание курсов, приведенные далее, описывают подходы к разработке программ и курсов и их преподаванию, однако они не являются ни инструктивными, ни исчерпывающими. При этом мы настоятельно рекомендуем, чтобы учреждения использовали данные, приведенные в этой главе, в качестве основы при разработке учебного плана, так как единая схема учебных планов в различных учреждениях принесет пользу как минимум трем группам: 1) студентам, решившим сменить учебное учреждение, 2) работодателям, которые хотели бы определить уровень знаний студентов, 3) разработчикам учебных материалов (авторам учебников и др.).

Необходимо отметить, что учреждениям, принимающим за основу учебного плана приведенные ниже образцы, рекомендуется предварительно проанализировать собственные потребности и адаптировать соответствующим образом учебный план. Локальные вопросы, изменяющиеся от учреждения к учреждению, включают:

1) начальную подготовку поступивших студентов;
2) наличие и компетентность профессорско-преподавательского состава учреждения;
3) общую культуру и цели вуза;
4) дополнительный материал, который учебное заведение считает нужным включить в программу обучения.

Начальной точкой в разработке программы должно стать определение подробного перечня желаемых результатов обучения студентов (см. главу 3).

Взаимосвязь с CCCS

Документ CC2001 Computer Science volume (CCCS) [ACM 2001] содержит перечень рекомендаций по построению университетских программ в области информатики. Несмотря на то, что программа подготовки бакалавров по программ-

ной инженерии отличается от соответствующей программы по информатике, обе специальности имеют много общего, особенно на уровне вводных курсов. По мере необходимости мы будем ссылаться на описания соответствующих курсов CCCS и покажем, как они могут быть адаптированы к программной инженерии. Это важно для тех многочисленных учебных заведений, которые выдают дипломы как по специальности «Информатика», так и по программной инженерии.

Как разрабатывалась данная глава

Чтобы разработать примеры учебных планов, подкомитет специалистов, работающих на общественных началах, разработал первый вариант документа. Затем последовало множество итераций, в которых изменения делались в основном членами организационного комитета на основе информации, полученной в результате различных семинаров. Члены комитета начали с SEEK, CCCS и анализа 32 программ подготовки бакалавров в Северной Америке, Европе и Австралии. Основным методом разработки учебного курса было определение, какие темы SEEK могут быть покрыты учебными курсами CCCS. Следующим важным шагом была разработка способов распределения оставшегося материала по связанным между собой курсам по программной инженерии, руководствуясь существующими учебными программами. Необходимо отметить, что большинство существующих учебных программ для бакалавров в действительности не охватывают весь объем SEEK, поэтому предложения не соответствовали в целом ни одной из программ.

После создания проекта данного документа как минимум один университет включил в учебный план большое количество описанных в данном документе курсов; обратная связь по данному эксперименту использовалась для усовершенствования приведенных здесь курсов.

6.1. Схема кодирования учебных курсов

В данном документе мы использовали следующую схему кодирования курсов:
XXnnn,
где:

XX – это одно из следующих значений:

CS – курс заимствован из CCCS;
SE – курс по программной инженерии, определенный в данном документе;
NT – нетехнический курс, определенный в данном документе;
MA – курс по математике, определенный в данном документе.

nnn – идентификационный номер, в котором:

- первая цифра определяет самый ранний год (из четырехлетнего периода), в течение которого обычно читается данный курс;

- вторая цифра разделяет курсы на широкие подкатегории внутри SE
  0 – означает, что курс является широким по охвату материала, покрывая многие области знаний SEEK;
  1 – означает, что в курсе делается упор на проектирование или на основы компьютинга, связанные с проектированием;
  2 – означает, что курс направлен на углубленное изучение процессов.
- третья цифра служит для различения курсов, которые в противном случае имели бы одинаковые номера.

Если не указано обратное, все курсы являются 40-часовыми стандартными курсами Североамериканской модели. Как обсуждалось ранее, это не означает, что материал необходимо преподавать именно в течение 40 часов, однако объем изучаемого материала должен быть эквивалентен объему традиционного курса, включающего 40 часов лекций плюс вдвое больше времени на самостоятельное изучение, лабораторные работы, индивидуальную работу с преподавателем, экзамены и др.

Идентификаторы курсов будут иметь различные виды затенения, шрифт, маркировку, что позволит различать категории курсов.

Первая категория курсов обычно преподается на ранних стадиях обучения и представляет собой базовый вводный материал. Данные об отдельных курсах данной категории и последовательность их преподавания можно представить в следующем виде:

Вводные курсы по SE/CS – начало первого года обучения
Вводные курсы CCCS по информатике
Курсы по основам математики

Вторая категория курсов большей частью содержит основной материал SEEK по программной инженерии. Более подробная информация приведена в разделе 6.3.

Базовые курсы по программной инженерии
Дипломный проект

Следующая группа учебных курсов является важной частью учебного плана, однако не содержит ни вводного материала, ни материала по программной инженерии. Более подробная информация о подобных курсах приведена в разделе 6.4.

Промежуточные базовые курсы по информатике

Обязательные нетехнические курсы

Следующие категории курсов являются выборочными (необязательными), по крайней мере, в некоторых учебных заведениях, хотя в других могут быть и обязательными. Дополнительная информация содержится в разделе 6.4.

Математические курсы, не входящие в основной набор курсов по SE

Технические курсы по выбору (SE/CS/IT/CE), не входящие в основной набор курсов по SE

Научные курсы/курсы по инженерии, раскрывающие темы, не входящие в SEEK

Общие нетехнические курсы

Без ограничений (--)

Последняя категория используется в тех случаях, когда в учебной программе выделено время для дополнительных курсов, однако конкретные темы курсов еще не определены.

6.2. Вводные последовательности курсов по программной инженерии, информатике и математике

В программах подготовки бакалавров существует несколько подходов введения в область программной инженерии для студентов первых полутора лет обучения. В данном разделе мы кратко опишем различные последовательности курсов и сами курсы. Вначале опишем последовательности изучения вводного материала по компьютингу и далее перейдем к описанию примеров последовательности изучения математических тем. Полные данные относительно новых учебных курсов, включая общее описание календарного плана обучения, предварительных требований к слушателям (prerequisites), задачи обучения, учебные модули, привязку к SEEK и другую информацию, приведены в приложении А. Приложение А содержит также привязку к SEEK курсов, заимствованных из CCCS.

Различие между двумя вводными последовательностями курсов по компьютингу состоит в том, что можно начать обучение студентов либо сразу с курсов, посвященных программной инженерии, либо с изучения в первый год основ информатики с переходом к программной инженерии только на второй год обучения. На сегодняшний день не существует свидетельств явного превосходства одного из

этих подходов над другим. Подход, ориентированный на информатику, в данный момент наиболее распространен и имеет веские причины таким и оставаться. Подход, ориентированный на программную инженерию, применяется в некоторых учебных заведениях для того, чтобы сразу познакомить студентов с изучаемым предметом. Ниже перечислены аргументы за и против обоих подходов:

Аргументы в пользу подхода, ориентированного на программную инженерию:

- Студенты с самого начала учатся мыслить как программные инженеры, концентрировать внимание на решаемой проблеме, изучать требования и особенности проектирования перед созданием кода, думать о процессах, работать по итеративной схеме и использовать преимущества других методов программной инженерии. Другими словами, они с самого начала вырабатывают в себе привычку думать обо всех деталях, необходимых для разработки больших систем.
- Менее вероятно, что у студентов выработается привычка мыслить только в терминах программного кода или о программном коде как о цели, а не средстве достижения цели. Существует мнение, что такой стиль мышления позднее трудно изменить и что далее он приводит к скептическому отношению студентов к основным принципам программной инженерии. Хорошо спланированный подход, ориентированный на информатику, поможет избежать этого, однако некоторые специалисты считают, что подход, ориентированный на программную инженерию, поможет избежать указанного недостатка с большей вероятностью.
- Преподавание программной инженерии на ранних этапах способствует тому, что студенты с самого начала чувствуют, что обучаются выбранной специальности.

Аргументы в пользу подхода, ориентированного на информатику:

- Программирование является основополагающим навыком для всех программных инженеров, однако для его совершенствования требуется длительная практика. Чем раньше и дольше студенты будут программировать, тем лучшими специалистами они смогут в дальнейшем стать. Некоторые могут отрицать важность программирования для программных инженеров, однако разработчики данного документа пришли к общему мнению, что это необходимый навык.
- Студенты, мало знакомые с компьютерами и программированием, могут не понять концепции SE, изучая их с первого года обучения, или же прийти к мнению, что данные концепции им мало о чем говорят.
- Существует множество учебников, предназначенных для стандартных курсов первого года обучения для CS-подхода и всего несколько учебников, которые действительно разработаны для SE-подхода. Поэтому обучение по методу SE может потребовать от преподавателей самостоятельного написания большей части необходимого им материала.

- Так как многие учреждения выдают дипломы как в области SE, так и CS, они могут захотеть преподавать определенные курсы студентам обоих направлений одновременно, чтобы снизить нагрузку на персонал.
- Во многих учреждениях не хватает специалистов в области SE. Существующие же специалисты по SE необходимы для преподавания более углубленных курсов. Перевод их на обучение студентов первого курса может сказаться на качестве более поздних курсов по SE.
- Большинство возможностей работы, доступных студентам первого года обучения, требует наличия навыков программирования. Работодатели вряд ли доверят студентам проектирование или работу с требованиями, пока те не приобретут достаточный для этого опыт. Таким образом, на первом году обучения необходимо делать упор на получения навыков программирования.

В обоих подходах есть свои положительные стороны и мало убедительных примеров, раскрывающих подходы только с «плохой» или только с «хорошей» стороны. Золотая середина состоит в том, что оба подхода действительно включают часть материала «противоположной стороны». В основной набор курсов первого года обучения по CCCS входит информация по SE, в то время как предлагаемые нами курсы первого года обучения SE согласно подходу «сначала SE» включают основы реализации, хотя и не так подробно, как курсы CS.

При этом предполагается, что к тому времени, как студенты завершат серию вводных курсов любой из последовательностей, они изучат одни и те же темы.

6.2.1. Серия вводных курсов по компьютингу A: изучение программной инженерии с первого года обучения

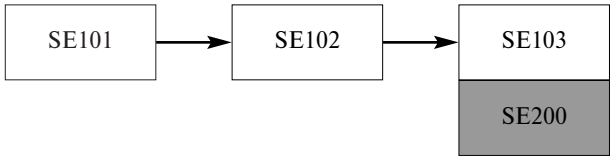
Данная последовательность предусматривает два курса в течение первого года обучения – SE101 и SE 102 (описываются ниже), которые знакомят с программной инженерией в совокупности с элементами программирования и другими концепциями информатики. Эти курсы отличаются от традиционных вводных курсов по информатике по двум основным признакам:

1) по причине более глубокого введения в область программной инженерии меньше времени выделяется на выработку навыков программирования;

2) тема инженерии является центральной темой данного курса. Таким образом, результат от включения нескольких дополнительных часов, формально посвященных программной инженерии, многократно усиливается благодаря акцентированию внимания на применении основ программной инженерии при решении всех задач по программированию.

В течение второго года обучения студенты слушают курсы CS103 и SE200, подготавливающие их к изучению серии промежуточных курсов, описанных в разделе 6.3. Совместное изучение курсов CS103 и SE200 завершает формирование основных знаний по компьютингу и навыков программирования у студентов по

данному учебному плану. Курс SE200 содержит некоторый материал по программированию, обычно включаемый во вводные курсы по компьютингу, но не включенный в курсы SE101 и SE102. Курсы CS103 и SE 200 могут читаться как параллельно, так и последовательно. Однако для наиболее эффективного распределения времени рекомендуется читать их одновременно.



Ниже приведено краткое описание упомянутых курсов.

SE101. Программная инженерия и компьютинг

Первый курс по программной инженерии и компьютингу для студентов, изучающих программную инженерию и не имеющих знаний по информатике уровня вуза. Знакомит с основными концепциями программирования, а также с базовыми концепциями программной инженерии.

SE102. Программная инженерия и компьютинг II

Второй курс по программной инженерии, углубляет знание концепций данной области и продолжает обучение основам информатики.

SE200. Программная инженерия и компьютинг III

Продолжает знакомство с концепциями программной инженерии и компьютинга.

CS103. Алгоритмы и структуры данных

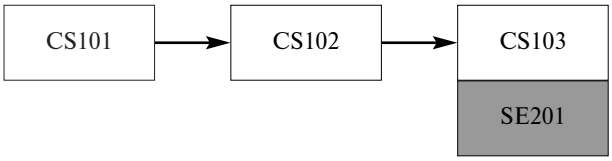
Можно использовать любой вариант курса CS103 из тома CCCS (например, основанный на императивном программировании или на объектно-ориентированном программировании). Обычно данный курс требует предварительного знакомства с CS102, но в нашем случае предшествующим курсом является SE102. Из описания, содержащегося в CCCS:

«Основываясь на познавательной базе курсов $CS101_I$-102_I, данный курс знакомит студентов с понятиями алгоритмов и структур данных. Темы курса включают в себя такие вопросы, как рекурсия, философия объектно-ориентированного программирования, базовые структуры данных (включая стеки, очереди, связанные списки, хэш-таблицы, деревья и графы), основы анализа алгоритмов и введение в принципы трансляции».

6.2.2. Серия вводных курсов по компьютингу B: Введение в область программной инженерии на втором году обучения

В данном случае студент начинает обучение с одной из последовательностей начальных курсов по информатике, описанных в документе CCCS для специаль-

ностей CS. Специализация в области программной инженерии начинается на втором году с изучения курса SE201, который можно читать одновременно с третьим курсом CS.



CCCS предлагает на рассмотрение несколько вариантов вводных курсов CS. Можно использовать любой из них, однако курсы с ориентацией на императивное программирование (индекс I) и курсы с ориентацией на объектное программирование (индекс O) считаются наиболее подходящими для преподавания основ программной инженерии. CS103 описан в предыдущем разделе; версии первых двух курсов CS с ориентацией на императивное программирование, а также SE201-int кратко описаны ниже и в Приложении A. Необходимо заметить, что CS201 и CS102 содержат большей частью основы компьютинга, предусмотренные SEEK, а также небольшую часть материала по программной инженерии из других областей знаний SEEK. Если даже включить в эти курсы основы программной инженерии, не ожидается, что в задачах преподавания программирования будет сделан большой упор на практике по программной инженерии.

CCCS предусматривает «сжатое» введение в информатику, согласно которому курсы CS101, CS102, CS103 объединены в последовательность из двух курсов – CS111 и CS112. При использовании данных курсов для преподавания программной инженерии раскрытия тем SEEK будет недостаточно, если слушающие курс студенты не будут иметь предварительных знаний в области CS или же если в другие курсы не будут добавлены дополнительные темы CS.

Курс $CS101_I$. Основы программирования

Данный курс является стандартным вводным курсом в информатику, использующим подход, ориентированный на императивное программирование. Описание курса из CCCS имеет следующий вид:

«Курс объясняет основные понятия процедурного программирования. Темы включают типы данных, управляющие структуры, функции, массивы, файлы и механизмы запуска, тестирования и отладки. Курс также содержит введение в исторический и социальный контекст компьютинга и обзор информатики как научной дисциплины».

$CS102_I$. Парадигма объектно-ориентированного программирования

Это второй курс из стандартной серии вводных курсов по информатике. Описание курса из CCCS имеет следующий вид:

«Курс знакомит студентов, освоивших парадигму процедурного программирования, с концепциями объектно-ориентированного программирования. Курс начинается с обзора управляющих структур и типов данных, уделяет

особое внимание структурированным типам данных и обработке массивов. Далее курс знакомит с парадигмой объектно-ориентированного программирования, фокусируясь на определении и использовании классов и основ объектно-ориентированного проектирования. Остальные темы курса содержат обзор основных принципов языков программирования, упрощенный анализ алгоритмов, основные методы поиска и сортировки, а также знакомит с некоторыми вопросами программной инженерии».

Курс SE201. Введение в программную инженерию

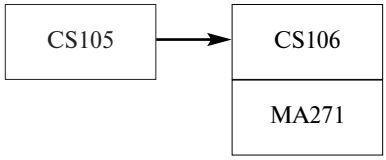
Основной курс, содержащий базовые принципы и концепции программной инженерии и закладывающий прочный фундамент для последующего изучения многих описанных ниже курсов. Он глубоко раскрывает важнейшие термины и концепции программной инженерии. По окончании данного курса студенты будут владеть основными навыками моделирования и проектирования, в частности, с использованием UML. Они также будут понимать основы работы с требованиями, программной архитектуры и тестирования.

6.2.3. Серия вводных курсов по математике

Дискретная математика является фундаментом для всего компьютинга, включая программную инженерию. Она столь же важна для программной инженерии, как и математический анализ для остальных инженерных специальностей. Статистика и эмпирические методы также имеют огромное значение для программной инженерии.

Курсы по основам математики покрывают темы FND.mf и частично FND.ef, предусматриваемые SEEK: дискретная математика, теория вероятности, статистика и эмпирические методы. Курсы CS105 и CS106 заимствованы из CCCS. В связи с тем, что специальность CS не содержит курса, раскрывающего некоторый материал, включенный в SEEK, был разработан новый курс MA271, включающий основы статистики и эмпирических методов.

Рекомендуется изучать данные курсы, начиная с первого учебного года, хотя это и не обязательное требование. Указанный материал необходим для некоторых промежуточных курсов по программной инженерии, описанных в следующем разделе.



Курс CS105. Дискретные структуры I

Данный курс является стандартным первым курсом по дискретной математике. При преподавании материала необходимо наглядно демонстрировать примеры его использования при проектировании программного и аппаратного обеспечения. Описание курса в документе CCCS имеет следующий вид: «Курс знакомит с основами дискретной математики и методами их использования в информатике. Основная задача курса – формирование прочной теоретической основы, необходимой для дальнейшей работы. Включает в себя следующие темы: функции, отношения, множества, простые методы доказательства, Булеву алгебру, логику высказываний, цифровую логику, элементарную теорию чисел и основы счета».

Курс CS106. Дискретные структуры II

Данный курс является вторым стандартным курсом по дискретной математике. Описание курса в документе CCCS имеет следующий вид:

«Данный курс продолжает изложение дискретной математики, начатое в курсе CS105. Темы данного курса включают в себя логику предикатов, рекуррентные соотношения, графы, деревья, матрицы, вычислительную сложность, элементарную вычислимость и дискретную вероятность».

Курс MA271. Статистика и эмпирические методы

Прикладная теория вероятности и статистика в контексте информатики. Проектирование экспериментов и анализ результатов. Обучение происходит с использованием примеров из программной инженерии и других дисциплин компьютинга.

6.3. Последовательности основных курсов по программной инженерии

В данном разделе мы представляем две последовательности, каждая из которых содержит 6 промежуточных курсов по программной инженерии. Мы также представляем дипломный проект. Подробное описание новых курсов, включая формальный календарный план, требования к слушателям, задачи обучения, преподаваемые модули, отображение на SEEK и другие материалы, находится в приложении A.

Ни одна из данных последовательностей не определена полностью (т.е. ни в одной из них все 40 часов не распределены по темам), что позволяет учебным учреждениям и преподавателям легко адаптировать их к своим потребностям.

Причиной разбиения материала на два блока является следующее:

- В некоторых учебных заведениях могут существовать курсы, соответствующие одному из блоков, которые желательно повторно использовать как можно большее количество раз. Например, в блоке I находится курс по требованиям, а блок II распределяет данный материал по другим курсам. С другой стороны, блок II содержит курс по тестированию, в то вре-

мя как в блоке I содержится курс, покрывающий как вопросы тестирования, так и вопросы качества.

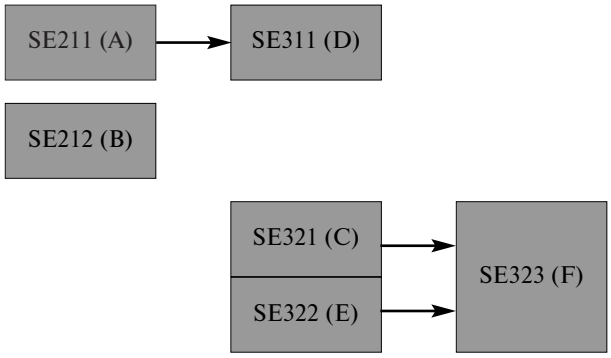
- Могут существовать индивидуальные предпочтения или предпочтения вузов в организации материала тем или иным способом. Например, некоторые предпочитают преподавать отдельный курс по формальным методам (блок II), другие – нет (блок I).

Не имеет значения, какой из блоков выбран, так как охват основных тем SEEK в итоге будет равноправным. Несмотря на это, покрытие обязательных и факультативных тем, так же как и добавленных вузом, будет отчасти отличаться.

Обе последовательности, состоящие из 6 курсов, имеют в качестве требований к своим слушателям SE201-int или SE 200 и обычно начинаются на втором году обучения. Также обе последовательности содержат SE212. В каждой последовательности курсы отмечены символами (A), (B) ... (F). Эти буквы используются в шаблонах курсов, обсуждаемых в главе 6.5, и обозначают позиции в расписании (slots), на которые могут быть помещены указанные курсы.

Отступ от левого края означает, что курс не следует включать в учебный план слишком рано, поскольку он требует определенной подготовки, но точно указанных требований к слушателям нет.

6.3.1. Пакет базовых курсов по программной инженерии I



Далее следуют названия и краткие описания курсов в этом блоке.

SE211. Разработка программного обеспечения

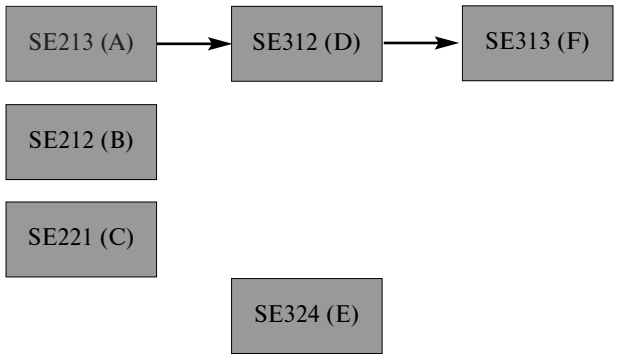
Покрывает вопросы детального проектирования, включая формальные подходы.

SE212. Подход программной инженерии к человеко-машинному взаимодействию

Покрывает различные темы, связанные с проектированием и оцениванием пользовательского интерфейса, а также некоторые основы психологии. Данный курс также находится в блоке II основных знаний по программной инженерии.

SE311. Проектирование и архитектура программного обеспечения
Углубленное изучение темы проектирования программного обеспечения, в том числе аспектов, связанных с распределенными системами и архитектурой программного обеспечения.
SE321. Обеспечение качества и тестирование программного обеспечения
Широкое освещение тем, связанных с качеством программного обеспечения и тестированием программного обеспечения.
SE322. Анализ требований к программному обеспечению
Широкое освещение темы требований к программному обеспечению в применении к его различным типам.
SE323. Управление программными проектами
Углубленный курс по управлению проектами. Предполагается, что данный курс будет прослушан студентами, уже имеющими достаточно широкое и глубокое понимание других аспектов программной инженерии.

6.3.2. Пакет базовых курсов по программной инженерии II



Отметим, что курс SE212-hci уже обсуждался в контексте пакета I. Главное отличие между данным пакетом и предыдущим в следующем:

- Данный блок объединяет весь материал по формальным методам в один курс SE313, представляя этот материал по плану позже, чем пакет I.
- Материалы по процессу, управлению и качеству организованы по-разному.
- Материал по проектированию рассматривается по принципу «сверху вниз», начиная с архитектурных тем.

SE213. Проектирование и архитектура больших программных систем
Моделирование и проектирование широкомасштабных, эволюционирующих систем; управление и планирование разработки таких систем, включая обсуждение управления конфигурациями и архитектурой программного обеспечения.

SE221. Тестирование программного обеспечения
Подробный курс по всем аспектам тестирования, а также по другим аспектам верификации и аттестации, включая определение тестируемых требований, рецензирование и обеспечение качества продукта.
SE312. Детальное проектирование программного обеспечения
Методы детального проектирования и разработки, включая формальные подходы. Детальное проектирование для обеспечения эволюционирования программного обеспечения.
SE324. Процесс разработки и управление разработкой программного обеспечения
Процесс разработки программного обеспечения в общих чертах; процессы управления и работы с требованиями; процессы эволюции; процессы управления качеством; управление персоналом проекта; планирование проекта.
SE313. Формальные методы программной инженерии
Подходы к проектированию и разработке программного обеспечения, использующие математический аппарат для достижения высоких уровней качества. Математические основы формальных методов; формальное моделирование; аттестация формальных моделей; формальный анализ проектирования; преобразование программ.

6.3.3. Дипломный проект по программной инженерии

Как уже обсуждалось, дипломный проект является исключительно важным в программе обучения программной инженерии. Дипломный проект предоставляет студентам возможность выполнить серьезный проект по программной инженерии, в котором они углубят свои знания в различных областях SEEK. Данный курс должен занимать целый учебный год (т.е. 80 лекционных часов). Он покрывает несколько часов различных тем SEEK, поскольку предполагается, что на протяжении курса студенты будут изучать некоторый материал самостоятельно и углубят свои знания в различных областях до уровня «а» по классификации Блума.

SE400 Дипломный проект по программной инженерии
Предоставляет студентам, работающим в группах, опыт разработки содержательного проекта, в котором они могут использовать изученный ранее по их программе материал, включая вопросы, связанные с требованиями, проектированием, человеческим фактором, профессионализмом и управлением проектами.

6.4. Завершая учебный план: дополнительные курсы

Вводные и основные курсы по программной инженерии, приведенные в двух последних разделах, охватывают большую часть необходимого материала, но для рассмотрения остается еще несколько категорий курсов. Подробное описание новых курсов, включая формальный календарный план, требования к слушателям, задачи обучения, преподаваемые модули, отображение на SEEK и другие материалы, находится в приложении A. Приложение A также содержит отображение на SEEK курсов, взятых из CCCS.

6.4.1. Курсы, покрывающие оставшийся обязательный материал

Промежуточные фундаментальные курсы по информатике (Int)

Промежуточные фундаментальные курсы по информатике являются курсами CCCS с номерами, начинающимися на 200, и покрывают большую часть оставшихся тем CMP.cf. Любой учебный план, охватывающий SEEK, должен содержать хотя бы два из них; во всех шаблонах следующего раздела включено по три выбранных курса, но это лишь один из вариантов возможных подходов. В некоторых учебных планах, не приведенных в данном документе, может потребоваться расширить промежуточный материал SEEK CMP.cf более чем на три курса.

Нетехнические (NT) обязательные курсы

Нетехнические обязательные курсы главным образом покрывают тему FND.ec и область PRF из SEEK – это инженерная экономика, навыки коммуникации и профессионализм. Несмотря на то, что имеется возможность сжать необходимый материал по SEEK до одного курса, мы показали пример распределения материала на три курса для более глубокого охвата.

NT272. Инженерная экономика

Это стандартный курс по инженерной экономике для многих университетов. Для изучения SEEK требуется относительно небольшая часть данного курса, однако желательно, чтобы изучение материала программными инженерами превышало минимум.

NT181. Групповая динамика и коммуникации

Навыки коммуникации и деловой корреспонденции (writing skills) высоко ценятся в программной индустрии и являются основой для построения успешной карьеры.

NT291. Профессиональная деятельность по программной инженерии

Профессиональная деятельность связана со знаниями, навыками и подходами, которыми должны обладать программные инженеры для профессиональной, ответственной и этичной работы в области программной инженерии. Подходящим альтернативным курсом является CS280 из тома CCCS.

6.4.2. Курсы, не входящие в SEEK

Некоторые курсы из приведенных ниже шаблонов учебного плана покрывают материал, выходящий за рамки SEEK. Данный материал включен в документ для содействия составителям учебных планов в разработке программ, покрывающих не только

SEEK. Определенное количество таких курсов необходимо для любой интересной и всесторонней программы по программной инженерии. Разработчики учебных планов и/или студенты имеют возможность делать свой собственный выбор, основываясь на потребностях учебного заведения, личных потребностях или на требованиях аккредитационных учреждений, которые стремятся расширить инженерную, научную или гуманитарную базу.

Математические курсы, не являющиеся основными для программной инженерии

Математическими курсами, не являющимися основными для программной инженерии, являются два типа математических курсов:

a) материал, который не является обязательным в программе обучения программной инженерии в соответствии с SEEK, однако по различным причинам требуется во многих учебных планах (например, математический анализ).

b) курсы по выбору.

Большинство университетов, особенно в Северной Америке, обычно преподают математический анализ с первого года обучения. SEEK не содержит курса по математическому анализу, поскольку он не используется программными инженерами, за исключением выполнения работы, связанной с конкретной предметной областью (например, для других инженеров, для ученых и для определенных заданий оптимизации), следовательно, не является необходимым для всех программ изучения программной инженерии. Как бы то ни было, существует определенное количество причин, по которым ряд программ будут содержать математический анализ: 1) предполагается, что математический анализ способствует развитию абстрактного мышления и математического мышления в целом; 2) множество статистических курсов в качестве требований к слушателям предъявляют математический анализ; 3) несмотря на то, что по работе это необходимо только небольшому проценту программных инженеров, математический анализ не так просто освоить прямо на рабочем месте.

Другими математическими предметами, которые присутствуют в учебном плане по программной инженерии, являются линейная алгебра и дифференциальные уравнения. (См. раздел 6.2.3 по математическим курсам (дискретная математика и теория статистики), являющимся частью основных курсов по программной инженерии.)

Технические курсы по выбору (SE/CS/IT/CE), не являющиеся основными для программной инженерии

Данные курсы покрывают технический материал вне рамок основных тем SEEK. Такие курсы могут быть обязательными в отдельных программах либо факультативными, выбираемыми студентами. Они могут покрывать темы в SEEK глубже, чем указано в SEEK, или могут покрывать материал, вообще не включенный в SEEK. В данной главе подробное описание этих курсов не приводится, однако для них отведены соответствующие часы в шаблонах учебных планов. Читатель может обратиться за более подробной информацией об этих курсах к другим томам Computing Curriculum, например, «Информатика», «Информационные системы» или «Проектирование аппаратных платформ».

Научные/инженерные курсы, которые посвящены темам, не входящим в SEEK

Сюда входит материал по физике, химии, электротехнике и т.д. Большинство программ изучения программной инженерии, особенно в Северной Америке, содержат такие курсы, в частности – физику.

Основной причиной включения научных курсов в программу является то, что они дают студенту опыт работы с научными методами и экспериментами. Аналогично, изучение других инженерных курсов расширяет восприятие студентами инженерии в целом. Изучение некоторых инженерных и научных курсов также поможет студентам, которые впоследствии захотят разрабатывать программное обеспечение в данных областях.

Курсы этой категории в данном документе подробно не описываются.

Общие нетехнические курсы (Gen ed)

К данному разделу относятся бизнес, социальные и гуманитарные науки, искусство и т.д. В большинстве программ некоторые из этих курсов будут обязательными, особенно в США, где существует традиция обязательного изучения гуманитарных наук. Также возможно включение в учебную программу последовательности нетехнических курсов по какому-то конкретному предмету (например, серии курсов по бизнесу).

6.5. Шаблоны учебного плана

В данном разделе мы представляем некоторые примеры шаблонов, демонстрирующие, как курсы, описанные в последних трех разделах, могут быть скомпонованы вместе с дополнительными неосновными курсами в программу обучения в университетах.

Все шаблоны должны рассматриваться только как примеры; они не являются руководствами (в отличие от SEEK). Они иллюстрируют подходы к комплектации тем SEEK в различных контекстах:

- Международный контекст.
- Контекст школы информатики или инженерии.

- Преподавание программной инженерии на первом году обучения или на втором.
- Деление академического год два семестра или три триместра.

Шаблон SE – Рекомендуемая общая структура

Year 1		Year 2		Year 3		Year 4	
Sem 1A	Sem 1B	Sem 2A	Sem 2B	Sem 3A	Sem 3B	Sem 4A	Sem 4B
Intro Computing Sequence			CS(Int)	CS(Int)	CS(Int)	SE400	SE400
CS105	CS106	*Calc 1*	*Calc 2*	MA271	SE	SE	*Tech elect*
NT		SE200/201	SE	SE	SE	*Tech elect*	*Tech elect*
		NT	SE	NT	*Tech elect*		

Остальная часть данной главы посвящена иллюстрации конкретных случаев применения Шаблона SE в различных контекстах.

Шаблон N2S-1 – Учебный год в Северной Америке, начало SE на 2 году обучения, семестры

Данный шаблон иллюстрирует вариант упорядочивания курсов, который построен в соответствии с потребностями многих университетов Северной Америки, придерживающихся семестровой системы. Для нескольких курсов названия не указаны – соответствующие позиции в расписании (slots) были введены с целью предусмотреть возможность адаптации. Два примера адаптации показаны ниже.

Техническое содержание шаблона начинается с CS101, CS102, и CS103. Шаблон также содержит SE201, изучаемый параллельно с CS103 (см. выше обсуждение данной последовательности); SE101, SE102, CS103, SE200 последовательность может быть заменена. После вводного курса SE201 (или SE200) студенты изучают один из блоков из шести курсов по SE, описанных выше, подробно покрывающих специфические области.

Поскольку промежуточные курсы по информатике достаточно гибкие, рекомендуется такой набор курсов CCCS, который покрывает соответствующие области SEEK.

Мы включили три нетехнических курса для покрытия важных областей SEEK. Мы рекомендуем начинать с курса по коммуникациям (например, NT181) на ранних этапах обучения и отложить курс по этике (например, NT291), как показано, до достижения студентами определенной профессиональной зрелости. Возможно много вариаций, включая объединение материала SEEK в один или два курса, вместо трех.

Мы поставили традиционный математический анализ (Calc 1 и Calc 2) на первый год, а математику программной инженерии – во второй семестр первого

курса. С педагогической точки зрения может прозвучать аргументация отложить математический анализ на более поздний срок; в любом случае преподавание математического анализа на первом курсе позволяет программам изучения программной инженерии быть согласованными с уже существующими программами по информатике и программной инженерии, а также гарантирует, что студенты, изучающие программную инженерию, слушают математический анализ с другими студентами той же возрастной группы.

Year1		Year 2		Year 3		Year 4	
Sem 1A	Sem 1B	Sem 2A	Sem 2B	Sem 3A	Sem 3B	Sem 4A	Sem 4B
CS101	CS102	CS103	CS (Int)	CS (Int)	CS(Int)	SE400	SE400
Calc 1	*Calc 2*	CS106	SE A	MA271	SE D	SE F	*Tech elect*
NT 181	CS105	SE201	SE212	SE C	SE E	*Tech elect*	*Tech elect*
--	--	NT 272	--	NT 291	*Tech elect*	--	--
--	--	--	--	--	--	--	--

Шаблон N2S-1c – На факультете информатики

Шаблон, показанный ниже, является типичным для программ по программной инженерии, которые могут читаться на факультетах информатики. Это адаптация Шаблона N2S-1, описанного выше. Такие программы получаются как результат преобразования программ по информатике или сосуществуют с ними.

Year1		Year 2		Year 3		Year 4	
Sem 1A	Sem 1B	Sem 2A	Sem 2B	Sem 3A	Sem 3B	Sem 4A	Sem 4B
CS101	CS102	CS103	CS220	CS226	CS270T	SE400	SE400
Calc 1	*Calc 2*	CS106	SE A	MA271	SE D	SE F	*Tech elect*
NT181	CS105	SE201	SE212	SE C	SE E	*Tech elect*	*Tech elect*
Physics	*Any science*	NT272	*Linear Alg*	NT291	*Tech elect*	*Tech elect*	*Tech elect*
Gen ed	*Gen ed*	--	*Gen ed*	*Gen ed*	*Gen ed*	*Gen ed*	*Gen ed*

Шаблон N2S-1e – На инженерном факультете

Программы инженерного факультета в Северной Америке обычно начинаются со строгой последовательности курсов математического анализа (три семестра), затем – теория вероятности и статистика, физика и химия. Вводные курсы по другим областям инженерии даются в течение первого года. Для программ по программной инженерии на факультетах электротехники и проектирования аппаратных платформ схемотехника и электротехника являются традиционными. Изучение программирования для инженеров всегда необходимо начинать в первый год. Вводная последовательность курсов по информатике часто является сжатой CS111, CS112 (CCCS), хотя мы придерживаемся последовательности из 3 курсов, описанной ниже, поскольку уверены, что так намного лучше для программных инженеров.

Year1		Year 2		Year 3		Year 4	
Sem 1A	Sem 1B	Sem 2A	Sem 2B	Sem 3A	Sem 3B	Sem 4A	Sem 4B
CS101	CS102	CS103	CS220	CS226	CS270T	SE400	SE400
Calc 1	*Calc 2*	CS106	SE A	MA271	SE D	SE F	*Tech elec*
NT181	CS105	SE201	SE212	SE C	SE E	*Tech elect*	*Tech elec*
Physics 1	*Physics 2*	NT272	*Lin Alg*	NT291	*Tech elect*	*Tech elect*	*Tech elec*
Chemistry	*Engineering*	*Calc 3*	*Gen ed*	*Gen ed*	*Gen ed*	*Gen ed*	*Gen ed*

Шаблон E-1 – Сжатая модель для страны, в которой математический анализ и естественные науки не являются обязательными для преподавания в вузе, или они изучаются в школе, или требуется менее общее образование

Некоторые страны, включая большую часть Великобритании, придерживаются системы средней школы, которая дает студентам более высокий уровень знаний по математике и естественным наукам. Такие системы также имеют тенденцию к концентрированному обучению конкретным предметам после окончания средней школы, выдвигая гораздо меньшие требования к общему образованию (гуманитарные науки и т.д.). Следующий шаблон демонстрирует вариант преподавания программной инженерии в таких средах.

Year1		Year 2		Year 3	
Term 1A	Term 1B	Term 2A	Term 2B	Term 3A	Term 3B
CS101	CS102	CS103	CS merged	SE400	SE400
CS105	CS106	MA271	SE D	SE F	*Tech elect*
NT181	SE201	SE A	SE E	*Tech elect*	*Tech elect*
NT272	NT291	SE C	SE212	*Tech elect*	*Tech elect*
--	--	--	--	--	--

Шаблон E-2 – Другая модель для страны, в которой математический анализ и естественные науки не являются обязательными

Данный шаблон также иллюстрирует использование SE101 и SE102, как и откладывание некоторых основных курсов программной инженерии до тех пор, пока студенты не достигнут определенного уровня зрелости.

Year1		Year 2		Year 3		Year 4	
Sem 1A	Sem 1B	Sem 2A	Sem 2B	Sem 3A	Sem 3B	Sem 4A	Sem 4B
SE101	SE102	CS103	SE200	SE A	SE212	SE D	SE F
CS overview	CS106	CS220	CS226	*Tech elect*	SE C	SE E	SE400
CS105	MA271	NT291	CS270T	*Tech elect*	*Tech elect*	SE400	*Tech elect*
NT181	NT272	--	--	--	--	--	--
--	--	--	--	--	--	--	--

Шаблон N3Q-1 – Учебный год в Северной Америке, начало на 3-м году обучения, триместры

Некоторые университеты Северной Америки придерживаются системы разбиения года по триместрам, с тремя триместрами вместо двух семестров. Следующий шаблон предоставляет такой вариант, подразумевая, что в каждом триместре преподаются четыре курса. Данный шаблон также иллюстрирует вариант, в котором основные курсы по программной инженерии откладываются до третьего года.

Year 1			Year 2		
Quarter 1A	Quarter 1B	Quarter 1C	Quarter 2A	Quarter 2B	Quarter 2C
CS101	*Calc 2*	CS102	CS 103	CS270T	CS226
Calc 1	*Chemistry*	*Calc 3*	CS220	CS106	*Math*
Physics 1	*Physics 2*	*Engineering*	CS105	NT291	*Gen ed*
Gen ed	NT181	*Gen ed*	*Math*	--	--

Year 3			Year 4		
Quarter 3A	Quarter 3B	Quarter 3C	Quarter 4A	Quarter 4B	Quarter 4C
SE201	SE A	SE D	SE400	SE400	SE400
SE212	SE C	SE E	SE F	*Tech elect*	*Tech elect*
MA271	*Tech elect*	*Gen ed*	*Tech elect*	*Gen ed*	*Gen ed*
NT272	--	--	*Gen ed*	--	--

Шаблон N1S – Модель США, в которой программная инженерия преподается с самого начала в составе курсов по информатике

Данная модель показывает использование начальной последовательности для первого года: SE101, SE102 и SE200.

Year1		Year 2		Year 3		Year 4	
Sem 1A	Sem 1B	Sem 2A	Sem 2B	Sem 3A	Sem 3B	Sem 4A	Sem 4B
SE101	SE102	CS103	CS270	CS220	SE D	CS226	SE400
Calc 1	*Calc 2*	SE200	SE212	SE A	SE E	SE400	*Tech elect*
CS105	CS106	*Physics 1*	MA271	SE C	*Tech elect*	SE F	*Tech elect*
Gen ed	*Psychology*	NT181	*Physics 2*	*Sci Elect*	NT291	*Gen ed*	*Gen ed*
Gen ed	*Gen ed*	*Gen ed*	*Sci Elect*	*Sci Elect*	--	NT272	--

Шаблон Jpn1 – Японский шаблон 1

Данный шаблон демонстрирует, как курсы могли бы преподаваться в Японии. Он основан на модели, представленной Японским Обществом по обработке

информации (IPSJ). Учебный план IPSJ был немного адаптирован, чтобы включить курсы из данного документа. Некоторые отличительные характеристики следующие: естественные и точные науки являются обязательными для слушателей, большое количество обязательных курсов по информатике, общеобразовательные курсы и дополнительные курсы по программированию читаются в основном в первый год. В Программе IPSJ на курсы может выделяться различное количество часов. Для упрощения, мы продемонстрировали пример программы, со стандартным количеством часов для курсов.

Year 1		Year 2		Year 3		Year 4	
Sem 1A	Sem 1B	Sem 2A	Sem 2B	Sem 3A	Sem 3B	Sem 4A	Sem 4B
Calc 1	*Calc 2*	CS	CS	CS	CS	SE400	SE400
CS111	CS112	CS	CS	SE C	SE E	*Tech elect*	NT181
CS extra	CS extra	CS	CS	SE D	SE F	*Tech elect*	*Tech elect*
CS105	CS106	CS	CS	NT291	NT272	--	--
Gen ed	*Gen ed*	MA271	SE A	*SysApp Spec*	*SysApp Spec*	--	--
Gen ed	*Gen ed*	SE201	SE212	*SysApp Spec*	*SysApp Spec*	--	--

Шаблон Aus1: Австралийская модель с четырьмя курсами в семестре

Данный шаблон демонстрирует модель, подходящую для Австралии. Он заимствован из учебного плана австралийского университета. Многие университеты Австралии склоняются к чтению четырех курсов в семестр, что дает студентам возможность изучить за один курс больше, чем если бы они слушали сразу пять или шесть курсов.

Как результат, 40-часовые курсы, обсуждаемые в этом документе, не подходят и должны быть адаптированы.

Некоторыми адаптациями являются следующие:

- Основы NT181 и NT272 покрыты в отдельном немного более длинном курсе.
- Материал по дискретной математике комбинирован в отдельный немного более длинный курс.
- Последовательности из шести курсов по программной инженерии не используются. Вместо этого включено пять обязательных курсов по SE сверх SE201. Два таких курса связаны с работой над проектом, обучение проходит в нелекционном формате
- Материал из SE323 и NT291 преподается в одном курсе.
- Некоторые из курсов по программной инженерии широко раскрывают темы SEEK, что достигается выбором частичных наборов технических факультативов.

Year1		Year 2		Year 3		Year 4		
Sem 1A	Sem 1B	Sem 2A	Sem 2B	Sem 3A	Sem 3B	Sem 4A		Sem 4B
CS101	CS102	CS220	CS103	CS	Team proj	SE400		SE400
Calc 1	*Lin. Alg*	CS270T	SE	SE	*Tech elect*	SE323	NT291	*Tech elect*
NT181/ NT 272	*Dig Logic*	SE201	Team proj	*Tech elect*	*Tech elect*	*Tech elect*		--
Intro EE	CS105 CS 106	MA271	--	--	--	--		--

Шаблон Isr1: Модель для Израиля

Данный шаблон сформирован из программы информатики израильского университета. Данная программа имеет большое количество предопределенных курсов по информатике. Для создания программы по программной инженерии мы заменили некоторые курсы по CS курсами по программной инженерии.

Year1		Year 2		Year 3		Year 4	
Sem 1A	Sem 1B	Sem 2A	Sem 2B	Sem 3A	Sem 3B	Sem 4A	Sem 4B
CS101	CS102	CS103	CS	SE A	SE D	SE400	SE400
Dig sys	CS	CS	CS	SE212	SE E	SE F	--
Calc 1	*Calc 2*	CS	CS	SE C	NT291	NT272	--
Lin. Alg	*Abst Alg*	MA271	CS	CS	--	--	--
NT181	Combinatorics	CS105 CS106	CS	CS	--	--	--

ГЛАВА 7. Адаптация к альтернативным средам

Учебные планы по программной инженерии не существуют сами по себе, а создаются в учебных заведениях с различными средами, целями и деятельностью. Учебные планы по программной инженерии должны допускать различные формы преподавания и быть применимыми в учебных заведениях различных типов.

В данном разделе будет говориться о двух главных группах «альтернатив». Первая группа – это нестандартные подходы к организации преподавательской деятельности, процесса передачи знаний студентам. Вторая – это новые подходы к организации работы учебного заведения.

7.1. Альтернативные среды обучения

В современном высшем образовании процесс обучения воспринимается прежде всего как лекционный процесс, в котором преподаватель проводит занятия в аудитории для большой группы студентов. Несмотря на то, что во многих вузах поддерживается индивидуальное обучение, доминирующим методом преподавания в большинстве высших учебных заведений является групповая работа в классе. Преподаватель проводит презентацию материала для аудитории в виде лекции, которой, возможно, сопутствует дискуссия. Лекции могут дополняться соответствующими лабораторными работами. Размер группы может варьироваться от менее 10 до более чем 500 студентов.

Преподавание компьютинга примечательно большим количеством экспериментов с различными способами преподавания материала. Возможно, это результат знания преподавателями возможностей технологий. Это также может быть связано с молодостью дисциплин компьютинга. Независимо от причин, множество публикаций представлено в сборнике SIGCSE, тезисах CSEE&T (Conference on Software Engineering Education & Training – Конференция по образованию и профессиональной подготовке в области программной инженерии), в тезисах FIE (Frontiers in Education – Конференция «Рубежи образования») и материалах других похожих форумов, в которых предлагались значительные модификации формата обычных лекций/семинаров, проводимых в аудитории. В качестве примеров можно привести лабораторные занятия, использование электронных досок и карманных компьютеров, обучение в процессе решения задач, ролевые игры, обучение путем участия в практической деятельности и различные студийные подходы, которые совмещают лабораторные работы, лекции и семинары. Как уже неоднократно подчеркивалось в данном документе, экспериментирование и исследование является обязательной частью любых учебных планов по программной инженерии. Необходимые изменения учебного плана тяжело реализовать в среде, которая не поддерживает экспериментирование и исследование. Учебный план по программной инженерии быстро устареет, если не прикладывать усилий к постоянному улучшению.

Один из последних направлений экспериментальной работы – это «дистанционное» обучение. Данный термин четко не определен. Он используется в ситуации, когда студенты физически расположены в различных местах во время проведения идущего по расписанию занятия. Также указанный термин применяется, когда студенты не только физически находятся в разных местах, но и отсутствует фиксированное расписание. Важно различать эти два случая. Также важно выделять другие случаи, например ситуации, в которых студенты не имеют возможности посещать занятия по расписанию.

7.1.1. Студенты физически находятся в различных местах

Задача обучения студентов, которые физически находятся в различных местах, имеет различные решения. В течение многих лет используется аудио и видеосвязь, и скоростной Интернет становится менее дорогим и все более доступным. Взаимодействие «преподаватель-студент» возможно после того, как все вовлеченные стороны четко усвоят, как правильно пользоваться технологией. Двухсторонняя видеосвязь делает это взаимодействие таким же естественным, как обычное взаимодействие в университетской аудитории. Для поддержки этого типа преподавания может быть задействована онлайн-база данных задач и примеров. Заменой аудиторному преподаванию могут служить web-ресурсы, e-mail и Интернет-чат. Выдача заданий может проводиться по электронной почте или с помощью непосредственного Интернет-соединения. Современная литература по компьютингу и университетские web-сайты содержат множество различных описаний методов дистанционного обучения.

Следует отметить, что полное решение проблемы преподавания курсов студентам, которые физически находятся в разных местах, – нетривиально, ведь любое спроектированное решение требует значительного планирования и соответствующей дополнительной поддержки. Возможно, появится возражение, что нет нужды предоставлять дополнительные средства поддержки при незначительном увеличении количества охваченных студентов, но, как показывает опыт, это не так.

Студенты, изучающие программную инженерию, нуждаются в опыте работы в командах. Работа географически распределенных студентов должна быть обеспечена соответствующим образом. Не стоит ожидать, что географически распределенная команда сможет сама эффективно выполнять работу с помощью электронной почты, чатов, блогов (blogs) и электронных групп новостей. Географически распределенные команды требуют дополнительного наблюдения и поддержки. Следует подумать об использовании видео- и телеконференций. Преподаватели, возможно, также захотят запланировать совещания команд, если позволяет расстояние. Начинающим студентам, в связи с отсутствием опыта работы в географически распределенной команде, необходимо уделить значительно больше внимания, нежели студентам, знакомым с таким методом работы.

Другая проблема, возникающая с географически распределенными студентами, – оценка их успеваемости. Необходимо найти организации для наблюдения за экзаменом и подтверждения личности экзаменуемого. Следует убедиться в том, что знания и навыки студента оцениваются несколькими различными способами. Слишком большое доверие к одному из методов (например, письменный экзамен) может сделать оценивание ненадежным.

7.1.2. Студенты занимаются в различное время

Некоторые образовательные учреждения имеют опыт проведения занятий для студентов-заочников, которые работают полный рабочий день. По причине выполнения своих рабочих обязанностей такие студенты часто не могут посещать обычные занятия. Хорошим подспорьем в такой ситуации будут записанные на видео лекции, копии конспектов и электронные копии презентаций. Web-сайт курса, электронная новостная группа и почтовая рассылка для обсуждения курса среди слушателей могут также служить поддержкой.

Существует такой тип обучения, который не подразумевает регулярных занятий в аудитории. Во многих учебных заведениях практикуются индивидуально планируемые (self-scheduled / self-paced) занятия. Занятия также проектируются для проведения целиком и полностью в web-среде. Разработано коммерческое программное обеспечение и программное обеспечение с открытым кодом для поддержки многих аспектов таких курсов. Опыт говорит о том, что разработка самостоятельных (self-paced) курсов для Web – очень дорогая деятельность, которая занимает много времени.

Студентам, которые не посещают регулярные занятия, все равно требуется опыт работы в команде. Для таких студентов также будут справедливыми замечания, которые сделаны для географически распределенных команд. Дополнительной проблемой является различный темп обучения студентов. Из-за разной скорости усвоения материала студентами может оказаться нереальным объединение в один модуль материалов и планов курса. Еще одной серьезной проблемой является проектирование индивидуально планируемых курсов. Тяжело координировать деятельность команды, участники которой работают с разной скоростью.

7.2. Учебные планы для альтернативных сред учебных заведений

7.2.1. Проблемы согласования

Проблемы согласования возникают в случаях, когда студенты, прослушав набор курсов в некотором учебном заведении либо в рамках некой учебной программы, хотят зачесть эти курсы в качестве прослушанных для продолжения образования в другом вузе и/или по другой учебной программе.

Если бы учебные планы по программной инженерии существовали автономно, то не было бы проблемы согласования. Однако в действительности это не

так. Программы обучения программной инженерии составлены в университетах, включающих множество институтов, школ, отделов, факультетов и специальностей. Некоторые средние общеобразовательные школы предлагают вузовский уровень преподавания, и студенты получают соответствующие зачеты и направления. Часто необходима сертификация удовлетворительного выполнения учебных планов, если студент посещает занятия в других кафедрах университета или в других учебных заведениях. Курсы по программной инженерии должны разрабатываться и читаться так, чтобы минимизировать проблему согласования. Это означает, что при разработке учебных планов следует учитывать внутренние и внешние по отношению к вузу среды.

7.2.2 Координация с другими университетскими курсами

Многие из курсов в учебных планах по программной инженерии могут также быть основными курсами в других учебных планах. Наличие вводного курса по информатике требуется в учебных планах по информатике, аппаратной инженерии и программной инженерии. Отдельные архитектурные курсы могут быть частью учебных планов по информатике, аппаратной инженерии, программной инженерии и электротехнике. Курс по управлению проектами может потребоваться при преподавании программной инженерии и управления информационными системами. Базовый уровень курсов по программной инженерии может преподаваться как часть программ по информатике или аппаратной инженерии. Во многих вузах максимально поощряется написание курсов, несущих двойную нагрузку.

Курсы, которые являются частью нескольких учебных планов, должны быть тщательно разработаны. Часто обстоятельства вынуждают включать все важное во все значимые дисциплины. Необходимо не поддаваться такому давлению, поскольку невозможно полностью удовлетворить интересы и желания всех. В курсах, которые служат двум хозяевам, неизбежно будут опущены темы, которые были бы освещены, если бы хозяин был один. Те, кто реализует учебные планы, должны осознавать, что лучшее – враг хорошего. Незначительная потеря материала при разработки курса для нескольких учебных планов значительно компенсируется опытом работы со студентами, обладающими альтернативными идеями и подготовкой. Действительно, можно привести пример, когда такой опыт является настолько важным в учебных планах по программной инженерии, что предпринимаются специальные усилия для того, чтобы один курс входил в несколько учебных планов.

7.2.3 Сотрудничество с другими вузами

В современном мире студенты получают высшее образование разными путями. В то время как большинство студентов получает образование в одном вузе,

имеется достаточное количество тех, кто обучается параллельно в нескольких. По разным причинам, многие студенты начинают свое обучение в одном вузе, а заканчивают в другом. При этом студенты могут поменять карьерные цели или специальность, могут перейти с гуманитарной программы на инженерную или научную программу, могут удовлетворять внутренние требования программы в одном вузе, могут заниматься производственной деятельностью или преодолевать финансовые, географические или личностные препятствия.

Учебный план по программной инженерии должен быть спроектирован таким образом, чтобы эти студенты смогли завершить начатое обучение без значительных задержек и повторений, путем признания сопоставимости курсов и выравнивания программ. Справедливым является перезачет предыдущей работы (на другом факультете или в другом вузе), если сравнение содержания курсов подтверждает их идентичность. Однако возникают проблемы, когда содержание курсов значительно отличается. С одной стороны никто не хочет, чтобы студент дважды сдавал один и тот же материал, также как никто не хочет, чтобы студент прослушал тот же самый материал дважды из-за того, что некоторые темы не были освещены в другом курсе. Профессорско-преподавательский состав не хочет задерживать обучение студентов из-за проблем с согласованием, поэтому наилучшими критериями, которые используются для перезачетов, является то, что от студентов ожидается:

1) своевременное приложение усилия для устранения любых несоответствий в знаниях,
2) прослушивание дальнейших курсов.

Наилучшим образом защищены интересы студентов в тех случаях, когда эквивалентность курсов заранее определена и существует соглашение по согласованию. Между многими вузами имеются формальные соглашения по переводу студентов на регулярной основе. Например, такие соглашения часто встречаются в США между вузами, которые готовят бакалавров, и средними специальными учебными заведениями. Другими примерами служат 2-3 соглашения в США между гуманитарными и техническими вузами. Эти соглашения позволяют студенту, проучившись три года в гуманитарном ВУЗе и два года в техническом, получить как степень бакалавра искусств (Bachelor of Arts), так и бакалавра наук (Bachelor of Science).

При формулировании соглашений по согласованию и разработке учебных планов важно рассматривать потенциальные требования к аккредитации. Аккредитация учебной программы распространяется на всех студентов только в том случае, если можно доказать, что студенты, которые переходят из других учебных заведений, изучили материал, аналогичный изученному студентами принимающего учебного заведения.

Европейская система зачетов является еще одной попыткой уменьшить проблему согласования.

7.3. Программы для учебных заведений по подготовке выпускников со степенью associate в США и общественных колледжей в Канаде

В США примерно половина выпускников, получивших степень бакалавра, начинает свое обучение в колледжах, выпускающих специалистов с дипломом младшего специалиста (associate). Поэтому важно наметить в общих чертах программу по обучению программной инженерии, которая может начаться в средних специальных учебных заведениях, особенно при расчете на последующий переход студентов в вузы (на 3-4-й год обучения). Вне зависимости от своих знаний при поступлении в колледж студенты должны целиком выполнить работу над курсом, набрав четко оговоренное количество баллов, продемонстрировав цельные знания и компетенцию для уверенности в успешной работе над курсом по программной инженерии в вузе. У некоторых студентов это может занять более двух лет, но вне зависимости от этого цель та же: разработать такую программу обучения, которая позволила бы подготовить студентов к обучению в высших учебных заведениях.

Далее приводится рекомендация по реализации программы подготовки по программной инженерии в колледжах. Студенты, которые выполнят эту программу, смогут обоснованно ожидать перевода в вуз для дальнейшего обучения. Хотя программа ориентирована на США, определенные колледжи в Канаде и других странах могут без труда использовать аналогичный подход.

Предложения по базовым техническим курсам для общественных колледжей (community colleges) Северной Америки

Для описания курсов по компьютингу и математических курсов, перечисленных ниже, следует смотреть отчет «Computing Curricula 2003: Guidelines for Associate-Degree Curricula in Computer Science» [ACM 2002].

Курсы по компьютингу:

Последовательность из трех курсов:

$CS101_I$ – Основы программирования

$CS102_I$ – Парадигма объектно-ориентированного программирования

$CS103_I$ – Структуры данных и алгоритмы

Альтернативная последовательность трех курсов:

$CS101_O$ – Введение в объектно-ориентированное программирование

$CS102_O$ – Абстракция объектов и данных

$CS103_O$ – Алгоритмы и структуры данных

SE201-int – Введение в программную инженерию для программных инженеров

Учебные заведения также могут принять решение построить учебный план по программной инженерии, основываясь на SE-курсах (SE101, SE102, CS103, SE200), описанных в главе 6 этого отчета.

Математические курсы:

CS105 – Дискретные структуры I

CS106 – Дискретные структуры II

Следующие курсы ориентированы на типичные университетские требования и не покрывают основной материал SEEK:

Математический анализ I

Математический анализ II

Также следует рассматривать требования вузов по подготовке бакалавров, например, для некоторых вузов необходима линейная алгебра и дифференциальные уравнения.

Курсы по физике и химии:

Два курса по физики или химии (laboratory science) для согласования с программами бакалавриата.

Рекомендация: два курса по физике или один по физике и один по химии.

Общее образование

Помимо базовых технических курсов по программной инженерии, студенты должны окончить общеобразовательные курсы для первого и второго годов обучения.

7.3.1. Специальные программы

Поскольку программная инженерия является очень молодой дисциплиной, то существует значительная потребность в определенных типах специальных программ. Кто-то хочет переквалифицироваться в новую область, другие, имея диплом о высшем образовании в родственной области, хотят получить второе высшее образование в программной инженерии. Учебные планы для таких программ должны принимать во внимание как предыдущее образование, полученное студентами, так и их карьерные цели.

Было бы глупо пытаться втиснуть все учебные планы по программной инженерии в короткую программу по переквалификации или одногодичную магистерскую программу. Такие усилия не помогают студентам достичь своих целей. Тем не менее в таких программах имеет смысл зафиксировать начальные требования к студентам, в которые входило бы наличие некоторого практического опыта. В этом случае студенты обычно достаточно хорошо мотивированы. У таких студентов есть опыт, который может заменить знания, получение которых является частью учебных планов вузов.

ГЛАВА 8. Внедрение и оценка программ обучения

8.1. Ресурсы и инфраструктура учебных планов

Как только учебный план сформирован, дальнейший его успех зависит в основном от трех ключевых элементов: профессорско-преподавательского состава, самих студентов и инфраструктуры среды обучения. В дополнение к ним, крайне значимой является регулярная поддержка со стороны индустрии.

8.1.1. Профессорско-преподавательский состав

Высокая квалификация преподавателей и другого персонала университета является, пожалуй, наиболее важным элементом успешности программы обучения. Для преподавания курсов, включенных в программу, ровно как и для любой другой предусмотренной учебными планами образовательной деятельности, необходимо иметь достаточное количество преподавателей; преподавательская и административная нагрузки должны позволять преподавателям участвовать в научной и профессиональной деятельности. Это важное условие успешности программы обучения таким динамичным дисциплинам, как компьютинг и программная инженерия.

Для преподавания программной инженерии необходимо наличие преподавателей, прошедших длительное обучение компьютингу с упором на программное обеспечение и достаточный практический опыт в области программной инженерии. Однако программная инженерия является относительно молодой дисциплиной, и сегодня чрезвычайно сложно набрать преподавателей, соответствующих традиционным академическим стандартам (ученая степень, способность к эффективному преподаванию, исследовательский потенциал) в совокупности с опытом работы в области программной инженерии [Glass 2003]. (Докторские программы по программной инженерии в США, например, были ведены только недавно). Необходимо поощрять и всячески поддерживать стремление преподавателей программной инженерии идти в ногу со временем, предоставляя им возможность проводить исследовательские работы, проходить интернатуру, получать необходимые консультации и т.п.

8.1.2. Студенты

Еще одним важным фактором успешности программы обучения являются студенты вуза. В вузе должны быть разработаны стандарты приема, позволяющие выбирать студентов, имеющих соответствующий уровень подготовки. Должны быть также разработаны процедуры и процессы отслеживания профессионального роста студентов в рамках программы обучения, что позволит определить, соот-

ветствует ли уровень знаний выпускников целям и желаемым результатам программы. Должны существовать метрики, соответствующие миссии учреждения и целям конкретной программы обучения, позволяющие направлять процесс обучения студентов для завершения программы в приемлемый период времени и оценивать, насколько уровень подготовки выпускников соответствует целям программы.

При обсуждении со студентами учебных планов и качества преподаваемого материала выявляется ценная информация, необходимая для оценки и анализа учебных программ. Участие студентов в профессиональных организациях и их деятельности расширит и углубит знания студентов.

8.1.3. Инфраструктура

Вуз должен предоставлять адекватную инфраструктуру и техническую поддержку. Сюда входят хорошо оборудованные лаборатории и учебные классы, удобные места для обучения и компетентный персонал лабораторий, способный оказать необходимую техническую помощь. Для того чтобы студенческие проектные группы могли эффективно работать, необходимо обеспечить помещения и оборудование для групповых собраний, инспекций, сквозного контроля программ, встреч с потребителями, регулярных совещаний команды и т.д. Необходимо предоставить в распоряжение студентам справочный материал и документацию, а также библиотеку, содержащую достаточный объем литературы по программной инженерии и другим дисциплинам компьютинга.

Поддержка состояния лабораторий и соответствующего набора программных средств может быть чрезвычайно сложной задачей вследствие динамичного и все ускоряющегося темпа развития программных и аппаратных платформ. Однако, как указывалось ранее в данном документе, необходимо помнить, что «студенты должны получать практический опыт, используя подходящие и современные средства».

Для обеспечения надлежащего администрирования программы по программной инженерии необходимо наличие эффективного руководства и персонала. Сюда входит соответствующий уровень консультирования студентов, службы поддержки и налаженная обратная связь с потенциальными клиентами-работодателями и выпускниками. Учебное заведение должно признать необходимость консультационной деятельности преподавателей и оказывать им всю необходимую административную поддержку.

Учебное заведение должно также обладать необходимыми финансовыми ресурсами для найма, поддержания необходимого уровня квалификации и удержания соответствующего преподавательского состава и другого персонала, поддержки соответствующей инфраструктуры и всей преподавательской деятельности в рамках программы обучения.

8.1.4. Участие индустрии

Еще одним важным элементом успешности учебного плана по программной инженерии является вовлечение и активное участие представителей индустрии. Промышленные консультационные советы и промышленно-учебные соглашения о сотрудничестве способствуют сохранению востребованности и обеспечению соответствия учебных планов текущим образовательным стандартам. Подобное сотрудничество может принимать различные формы: проведение консультаций, производственная интернатура для студентов и преподавателей, включение производственных проектов в учебный план, чтение лекций приглашенными представителями индустрии или зачисление их в штат преподавателей.

8.2. Вопросы оценки и аккредитации учебных планов

Чтобы поддерживать качественный уровень учебных планов, необходимо регулярно проводить их оценку. Широко распространено мнение, что оценка должна проводиться для получения формальной аккредитации. Рекомендации по составлению учебных планов, стандарты и критерии аккредитации предоставляются различными аккредитационными организациями по всему миру [ABET 2000, BCS 2001, CEAB 2002, ECSA 2000, King 1997, IEI 2000, ISA 1999, JABEE 2003]. В некоторых странах оценка учебных планов производится правительственными организациями по заранее определенной стандартной модели учебных планов, наборам стандартов к ним и соответствующих рекомендаций. В 1998 году рабочая группа, созданная совместно IEEE и ACM, разработала проект критериев аккредитации для учебных планов по программной инженерии [Barnes 1998], включающий рекомендации и требования для профессорско-преподавательского состава, учебного плана, лабораторных и вычислительных ресурсов, студентов, уровню поддержки со стороны учреждения и процедуры оценки эффективности программы. Что касается учебного плана, в проекте было сказано, что программа степени бакалавра в области программной инженерии должна включать приблизительно равные сегменты обучения *программной инженерии*, *информатике* и *инженерии*, в соответствующих *вспомогательных областях* и в изучении *углубленного материала*.

В общем случае процесс аккредитации означает периодическое внешнее рецензирование программ обучения, что обеспечивает соответствие минимальному набору критериев и стандартов аккредитационной организации. Одним из общепринятых подходов к оценке и аккредитации является «подход, ориентированный на результат», согласно которому вначале устанавливаются цели обучения и/или желаемые результаты; далее учебный план, управляющая организация и инфраструктура анализируются с точки зрения их соответствия выбранным целям.

В процессе оценки должны анализироваться цели программы обучения и желаемые результаты, содержание учебного плана и методы его преподавания, и

этот процесс является основным механизмом обратной связи, необходимым для постоянного улучшения программы обучения.

В дополнение к данному документу и указанным ранее аккредитационным организациям, существует множество источников, способствующих определению и оценке целей и желаемых результатов программы [Bagert 1999, Lethbridge 2000, Meyer 2001, Naveda 1997, Parnas 1999, Saiedian 2002; IWCSEA].

8.3. Программная инженерия в других дисциплинах компьютинга

Программная инженерия, конечно же, не существует сама по себе. Она находится в тесной взаимосвязи с другими областями науки и технологии, особенно с другими дисциплинами компьютинга. С одной стороны находятся ученые, а с другой – технология и технические специалисты. Центральное место занимает проектирование, являющееся отличительной чертой всех учебных инженерных программ.

В данном контексте специалисты по информатике в основном концентрируются на получении новых знаний в форме новых алгоритмов, структур данных, методов поиска данных, выявления новых принципов организации человеко-машинного взаимодействия, оптимизированных операционных систем и сетей, современных языков программирования и средств, которые могут использоваться для улучшения работы программных инженеров (а в данном случае – и компьютерных инженеров). Необходимо отметить, что в том CCCS включена глава под названием «Изменения в информатике как в дисциплине» и что существует большое количество точек зрения относительно информатики как дисциплины, а также стоит заметить, что необходимо различать существующую информатику и то, какой она может стать в ближайшем будущем – дисциплиной, изучающей теоретические основы и ограничения компьютинга. Девид Парнас [Parnas 99] затрагивает данный вопрос в высказывании: «инженер не может быть уверен, что продукт готов к использованию, пока ему не станут известны ограничения этого продукта». Подобные ограничения включают в себя технологические ограничения (ограничения доступных аппаратных, программных средств и средств проектирования), а также фундаментальные ограничения (теории вычислимости и сложности, теория информации и т.п.).

Прикладные и специализированные программы, такие как администрирование сетей и систем, все программы по инженерным технологиям, являются в некотором роде противоположными по отношению к информатике. Программная и компьютерная инженерия располагаются в центре спектра, фокусируясь на инженерном проектировании. Основная роль инженерного проектирования в рамках программной инженерии обсуждается в других частях данного документа. Программный инженер должен концентрироваться на понимании того, как использовать теоретические знания для решения практических задач.

Благодаря всеобъемлющей природе программного обеспечения, рамки вопросов программной инженерии могут быть значительно шире, чем рамки других направлений инженерии. В рамках определенной *предметной области* проектировщику необходимо основываться на особых знаниях и опыте для оценки большого количества возможных решений. Ему необходимо определить, какие стандартные компоненты могут быть использованы, а какие нужно разработать с нуля. Для принятия необходимых решений проектировщик должен иметь основные знания в соответствующих областях. В то время как области применения охватывают полный спектр производственных, правительственных и социальных задач, существует более короткий перечень конкретных областей применения, таких как научные информационные системы, – включая биоинформатику, астроинформатику, экоинформатику, микросистемы, аэронавтику и астронавтику и др.

Библиография по преподаванию программной инженерии

[Abelson 1985] Abelson H. and Sussman G. J. Structure and Interpretation of Computer Programs. Cambridge, MA: MIT Press, 1985.

[ABET 2000] Accreditation Board for Engineering and Technology, Accreditation policy and procedure manual, ABET Inc., November 2000 (http://www.abet.org/images/policies.pdf).

[ACM 1965] ACM Curriculum Committee on Computer Science. An undergraduate program in computer science-preliminary recommendations. Communications of the ACM, September 1965.

[ACM 1968] ACM Curriculum Committee on Computer Science. Curriculum '68: Recommendations for the undergraduate program in computer science. Communications of the ACM, March 1968.

[ACM 1978] ACM Curriculum Committee on Computer Science. Curriculum '78: Recommendations for the undergraduate program in computer science. Communications of the ACM, March 1979.

[ACM 1989] ACM Task Force on the Core of Computer Science. Computing as a Discipline. Communications of the ACM, January 1989.

[ACM 1998] ACM/IEEE-CS Joint Task Force on Software Engineering Ethics and Professional Practices, Software Engineering Code of Ethics and Professional Practice, Version 5.2, (http://www.acm.org/serving/se/code.htm), September 1998.

[ACM 1999] ACM Two-Year College Education Committee. Guidelines for associate-degree and certificate programs to support computing in a networked environment, The Association for Computing Machinery, September 1999.

[ACM 2001] ACM/IEEE-Curriculum 2001 Task Force, Computing Curricula 2001, Computer Science , December 2001 (http://www.computer.org/education/cc2001/final/index.htm),

[ACM 2002] ACM/IEEE-Curriculum 2001 Task Force, Computing Curricula 2003: Guidelines for Associate-Degree Curricula in Computer Science, December 2002 (http://www.acmtyc.org/reports/TYC_CS2003_report.pdf).

[Andrews 2000] Andrews J.H. and Lutfiyya H.L. Experiences with a Software Maintenance Project Course. IEEE Transactions on Education, November 2000.

[APP 2000] Advanced Placement Program, Introduction of Java in 2003-2004, The College Board, December 2000 (http://www.collegeboard.org/ap/computer-science).

[Bagert 1999] Bagert D. et al. Guidelines for Software Engineering Education, Version 1.0, CMU/SEI-99-TR-032, Software Engineering Institute, Carnegie Mellon University, 1999.

[Barnes 1998] Barnes B. et al. Draft Software Engineering Accreditation Criteria. Computer, April 1998.

[Barta 1993] Barta B.Z., Hung S.L. and Cox K.R. (Eds.), IFIP Transactions A40, Software Engineering Education, Proceedings of the IFIP W.G. 3.4/SRIG-ET (SEARCC) International Working Conference, Hong Kong, September 1993, North-Holland, Amsterdam, 1993.

[Bauer 1972] Bauer F.L. Software Engineering. Information Processing, 71, 1972.

[BCS 1989a] British Computer Society and The Institution of Electrical Engineers, Undergraduate curricula for software engineers, London, June 1989.

[BCS 1989b] British Computer Society and The Institution of Electrical Engineers, Software in safety-related systems, London, October 1989.

[BCS 2001] British Computer Society, Guidelines On Course Exemption & Accreditation For Information For Universities And Colleges, August 2001 (http://www1.bcs.org.uk/link.asp?sectionID=1114).

[Beidler et al, 1985] Beidler J., Austing R. and Cassel L. Computing Programs in Small Colleges, Communications of the ACM, June 1985.

[Bennett 1986] Bennett W. A Position Paper on Guidelines for Electrical and Computer Engineering Education, IEEE Transactions in Education, August 1986.

[Bloom 1956] Bloom B.S. (Ed.) Taxonomy of educational objectives: The classification of educational goals: Handbook I, cognitive domain. Longmans, 1956.

[Bourque 2001] Bourque P.and Dupuis R. (Eds.). Guide to the Software Engineering Body of Knowledge, IEEE CS Press, 2001.

[Borstler 2002] Borstler J. et al. Teaching PSP: Challenges and Lessons Learned, IEEE Software, September/October 2002.

[Bott 1995] Bott F. et al. Professional Issues in Software Engineering, 2nd Ed., UCL Press, 1995.

[Brooks 95] Brooks F.P. The Mythical Man-Month, Essays on Software Engineering, Anniversary Edition, Addison-Wesley, 1995.

[Budgen 2003] Budgen D. and Tomayko J.E. Norm Gibbs and His Contribution to Software Engineering Education Through the SEI Curriculum Modules, Proceedings of the 16th Conference on CSEE&T, March 2003.

[Burnell 2002] Burnell L.J., Priest J.W. and Durrett J.R. Teaching Distributed Multidisciplinary Software Development, IEEE Software, September/October, 2002.

[Buxton 1970] Buxton J.N. and Randell B. (Eds.) Software Engineering Techniques, Report of a Conference Sponsored by NATO Science Committee (Rome, 27 31 October, 1969), 1970.

[Carnegie 1992] Carnegie Commission on Science, Technology, and Government, Enabling the Future: Linking Science and Technology to Societal Goals, Carnegie Commission, September 1992.

[Cheston 2002] Cheston G.A. and Tremblay J.-P. Integrating Software Engineering in Introductory Computing Courses, IEEE Software, September/October 2002.

[CEAB 2002] Canadian Engineering Accreditation Board, Accreditation Criteria and Procedures, Canadian Council of Professional Engineers, 2002 (http://www.ccpe.ca/e/files/report_ceab.pdf).

[COSINE, 1967] COSINE Committee, Computer Science in Electrical Engineering. Washington, DC: Commission on Engineering Education, September 1967.

[Cowling 1998] Cowling A. The First Decade of an Undergraduate Degree Programme in Software Engineering, Annals of Software Engineering, V. 6, PP. 61-90, 1998.

[CSAB 1986] Computing Sciences Accreditation Board, Defining the Computing Sciences Professions, October 1986 (http://www.csab.org/comp_sci_profession.html).

[CSAB 2000] Computing Sciences Accreditation Board, Criteria for Accrediting Programs in Computer Science in the United States, Version 1.0, January 2000 (http://www.csab.org/criteria2k_v10.html).

[CSTB 1994] Computing Science and Telecommunications Board, Realizing the Information Future, Washington DC: National Academy Press, 1994.

[CSTB 1999] Computing Science and Telecommunications Board, Being Fluent with Information Technology, Washington DC: National Academy Press, 1999.

[Curtis 1983] Curtis K.K. Computer manpower: Is there a crisis? Washington DC: National Science Foundation, 1983 (http://www.acm.org/sigcse/papers/curtis83)

[Cybulski 2000] Cybulski J.L. and Linden T. Learning Systems Design with UML and Patterns, IEEE Transactions on Education, November 2000.

[Davis 1997] Davis G.B. et al. IS'97 Model Curriculum and Guidelines for Undergraduate Degree Programs in Information Systems, Association of Information Technology Professionals, 1997 (http://webfoot.csom.umn.edu/faculty/gdavis/curcomre.pdf).

[Denning 1989] Denning P.J. et al. Computing as a Discipline, Communications of the ACM, January 1989.

[Denning 1992] Denning P.J. Educating a New Engineer, Communications of the ACM, December, December 1992.

[Denning 1998] Denning P.J. Computing the profession, Educom Review, November 1998.

[Denning 1999] Denning P.J. Our Seed Corn is Growing in the Commons, Information Impacts Magazine, March 1999 (http://www.cisp.org/imp/march_99/denning/03_99denning.htm).
[EAB 1983] Educational Activities Board, The 1983 Model Program in Computer Science and Engineering, Technical Report 932, IEEE Computer Society, December 1983.
[EAB 1986] Educational Activities Board, Design Education in Computer Science and Engineering, Technical Report 971, IEEE Computer Society, October 1986.
[EC 1977] Education Committee of the IEEE Computer Society, A Curriculum in Computer Science and Engineering, Publication EHO119-8, IEEE Computer Society, January 1977.
[ECSA 2000] Engineering Council Of South Africa, Policy on Accreditation of University Bachelors Degrees, August 2000 (http://www.ecsa.co.za).
[Fairley 1985] Fairley R. Software Engineering Concepts, McGraw-Hill, 1985.
[Finkelstein 1993] Finkelstein A. European Computing Curricula: A Guide and Comparative Analysis, Computer Journal, V. 36, № 4, PP. 299-319, 1993.
[Fleddermann 2000] Fleddermann C.B. Engineering Ethics Cases for Electrical and Computer Engineering Students, IEEE Transactions on Education, V. 43, № 3, PP. 284-287, August 2000.
[Ford 1994] Ford G. A Progress Report on Undergraduate Software Engineering Education, CMU/SEI-94-TR-11, Software Engineering Institute, Carnegie Mellon University, May 1994.
[Ford 1996] Ford G. and Gibbs N.E. A Mature Profession of Software Engineering, CMU/SEI-96-TR-004, Software Engineering Institute, Carnegie Mellon University, January 1996.
[Freeman 1976] Freeman P., Wasserman A.I. and Fairley R.E. Essential Elements of Software Engineering Education, Proc. of the 2nd International Conference on Software Engineering, IEEE Computer Society Press, 1976, PP. 116-122.
[Freeman 1978] Freeman P. and Wasserman A.I. A Proposed Curriculum for Software Engineering Education, Proc. of the 3rd International Conference on Software Engineering, Atlanta, 1978, PP. 56-62.
[Gibbs 1986] Gibbs N.E. and Tucker A.B. Model Curriculum for a Liberal Arts Degree in Computer Science, Communications of the ACM, 29(3):202-210, March 1986.
[Giladi 1999] Giladi R. An Undergraduate Degree Program for Communications Systems Engineering, IEEE Transactions on Education, V. 42, № 4, PP. 295-304, November 1999.

[Glass 2003] Glass R.L. A Big Problem in Academic Software Engineering and a Potential Outside-the-Box Solution, IEEE Software, V. 20, № 4, July/August 2003.

[Gorgone 2002] Gorgone J.T. et al. IS 2002: Model Curriculum for Undergraduate Degree Programs in Information Systems, published by the ACM, 2002.

[Hilburn 2002a] Hilburn T.B. Software Engineering Education: A Modest Proposal, IEEE Software, V. 14, № 4, November 1997.

[Hilburn, 2002b] Hilburn T.B. and Humphrey W.S. The Impending Changes in Software Education, IEEE Software, V. 19, № 5, September/October, PP. 22-24, 2002.

[Hilburn, 2003] Hilburn T.B., Sobel A.E.K., Hislop G.W. and Duley R. Engineering an Introductory Software Engineering Curriculum, Proceedings of the 16th Conference on CSEE&T, 99-106, March 2003.

[Hunter 2001] Hunter R. and Thayer R.H. (Eds.) Software Process Improvement, IEEE Computer Society, Los Alamitos, CA, 2001.

[IEI 2000] The Institution of Engineers of Ireland, Accreditation of Engineering Degrees, May 2000 (http://www.iei.ie/Accred/accofeng.pdf).

[ISA 1999] Institution of Engineers, Australia, Manual For The Accreditation Of Professional Engineering Programs, October 1999 (http://www.ieaust.org.au/membership/res/downloads/AccredManual.pdf).

[ISRI 2003] Institute for Software Research, International, PhD Program in Software Engineering, School of computer Science Carnegie Mellon University, 2003 (http://www-2.cs.cmu.edu/afs/cs.cmu.edu/project/isri/www/Design/phd.html).

[IEEE 1990] IEEE STD 610.12-1990, IEEE Standard Glossary of Software Engineering Terminology, IEEE Computer Society, 1990.

[IEEE 2001] Institute for Electrical and Electronic Engineers. IEEE code of ethics. Piscataway, NJ: IEEE, May 2001 (http://www.ieee.org/about/whatis/code.html).

[IEEE 2003] Certified Software Development Professional, IEEE Computer Society (http://www.computer.org/certification).

[JABEE 2003] Japan Accreditation Board for Engineering, Criteria for Accrediting Japanese Engineering Education Programs 2002-2003 (http://www.jabee.org/english/OpenHomePage/e_criteria&procedures.htm).

[Juristo 2003] Juristo N. Analysis of Software Engineering Degree Establishment in Europe, Keynote Address, 16th Conference on Software Engineering Education & Training, March 2003 (http://www.ls.fi.upm.es/cseet03/keynotes/Natalia_Juristo_CSEET03.pdf).

[Kelemen 1999] Kelemen C.F. (Ed.), Computer Science Report to the CUPM Curriculum Foundations Workshop in Physics and Computer Science. Report from a workshop at Bowdoin College, October 28-31, 1999.
[Kemper 1990] Kemper J. Engineers and Their Profession, Oxford University Press, 1990.
[King 1997] King W.K., Engel G. Report on the International Workshop on Computer Science and Engineering Accreditation, Salt City, Utah, 1996, Computer Society, 1997
[Koffmanl 1984] Koffman E.P., Miller P.L. and Wardle C.E. Recommended curriculum for CS1: 1984 a report of the ACM curriculum task force for CS1, Communications of the ACM, 27(10):998-1001, October 1984.
[Koffman 1985] Koffman E.P., Stemple D. and Wardle C.E. Recommended Curriculum for CS2, 1984: A Report of the ACM Curriculum Task Force for CS2, Communications of the ACM, 28(8):815-818, August 1985.
[Lee 1998] Lee E.A. and Messerschmitt D.G. Engineering and Education for the Future, IEEE Computer, 77-85, January 1998.
[Lethbridge 2000] Lethbridge T. What Knowledge is Important to a Software Engineer?, IEEE Computer, V. 33, № 6, PP. 44-50, May 2000.
[Lidtke 1999] Lidtke D.K. et al. ISCC '99: An Information Systems-Centric Curriculum '99, July 1999 (http://www.iscc.unomaha.edu).
[Lutz 2001] Lutz M.J. Software Engineering on Internet Time, Computer, 34, 5, 36, May 2001.
[Marciniak 1994] Marciniak J. (Editor-in-chief), Encyclopedia of Software Engineering, John Wiley & Sons, 1994.
[Martin 1996] Martin C.D. et al. Implementing a Tenth Strand in the CS Curriculum, Communications of the ACM, 39(12):75-84, December 1996.
[McConnell 1999] McConnell S. and Tripp L. Professional Software Engineering: Fact or Fiction? IEEE Software, V. 16, № 6, November/December, 1999, PP. 13-17.
[McDermid, 1991] McDermid J. (Ed.) Software Engineer's Reference Book, Butterworth-Heinemann Ltd, Oxford, England, 1991.
[Meyer 2001] Meyer B. Software Engineering in the Academy, IEEE Computer, 34, 5, PP. 28-35, May 2001.
[Mulder 1975] Mulder M.C., Model Curricula for Four-Year Computer Science and Engineering Programs: Bridging the Tar Pit, Computer, 8(12):28-33, December 1975.
[Mulder 1984] Mulder M.C. and Dalphin J. Computer Science Program Requirements and Accreditation-an Interim Report of the ACM/IEEE Computer Society Joint Task Force, Communications of the ACM, 27(4):330-335, April 1984.

[Mulder 1998] Mulder F. and van Weert T. Informatics in Higher Education: Views on Informatics and Non-informatics Curricula, Proceedings of the IFIP/WG3.2 Working Conference on Informatics (computer science) as a discipline and in other disciplines: What is in common?, Chapman and Hall, London, 1998.

[Myers 1997] Myers C., Hall T. and Pitt D. (Eds.) Proceedings of the First Westminster Conference: Professional Awareness in Software Engineering (PASE'96), London, February 1996 (Published in edited form as: The Responsible Software Engineer, Springer-Verlag, London, 1997, ISBN 3-540-76041-5).

[NACE 2003] National Association of Colleges and Employers. Job Outlook 2003. (http://www.naceweb.org)

[Naur 1969] Naur P. and Randell B. (Eds.) Software Engineering: Report on a Conference Sponsored by the NATO Science Committee, (7 – 11 October 1968), Brussels, Scientific Affairs Division, NATO, 1969.

[Naveda 1997] Naveda J.F and Lutz M.J. The Road Less Traveled: A Baccalaureate Degree in Software Engineering, Proceedings of 1997 Conference on Software Engineering Education & Training, April, 1997.

[Neumann 1995] Neumann P.G. Computer Related Risks, New York: ACM Press, 1995.

[Nordheden 1999] Nordheden K.J. and Hoeflich M.H. Undergraduate Research & Intellectual Property Rights, IEEE Transactions on Education, № 42(4), P. 233, 1999.

[NSF 1996] National Science Foundation Advisory Committee, Shaping the Future: New Expectations for Undergraduate Education in Science, Mathematics, Engineering, and Technology, Washington DC: National Science Foundation, 1996.

[NTIA 1999] National Telecommunications and Information Administration, Falling through the Net: Defining the Digital Divide, Department of Commerce, November 1999.

[Nunamaker 1982] Nunamaker Jr. J. F., Couger J. D. and Davis G.B. Information Systems Curriculum Recommendations for the 80s: Undergraduate and Graduate programs, Communications of the ACM, 25(11):781-805, November 1982.

[Oklobdzija 2002] Oklobdzija V.G. (Ed.) The Computer Engineering Handbook, CRC Press, 2002.

[OTA 1988] Office of Technology Assessment, Educating Scientists and Engineers: Grade School to Grad School, OTA-SET-377, U.S. Government Printing Office, June 1988.

[QAA 2000] Quality Assurance Agency for Higher Education, A Report on Benchmark Levels for Computing, Southgate House, 2000.

[Parnas 1999] Parnas D.L. Software Engineering programs Are Not Computer Science Programs, IEEE Software, November/December 1999, PP. 19-30.

[Paulk 1995] Paulk M. et al. The capability maturity model: Guidelines for Improving the Software Process, Reading, MA: Addison-Wesley, 1995.

[PMI 2000] Project Management Institute, Guide to the Project Management Body of Knowledge, PMI, 2000.

[Ralston 2000] Ralston A., Reilly E.D. and Hemmendinger D. (Eds.), Encyclopedia of Computer Science, 4th edition, Nature Publishing Group, London, England, 2000.

[Ramamoorthy 1996] Ramamoorthy C.V. and Thai W. Advances in Software Engineering, Communications of the ACM, 29, 10, 47-58, October, 1996.

[Richard 1999] Richard W.D. Taylor D.E. and Zar D.M. A Capstone Computer Engineering Design Course, IEEE Transactions on Education, V. 42, № 4, PP. 288-294, November 1999.

[Roberts 2001] Roberts E. and Engel G. (Eds.) Computing Curricula 2001: Computer Science, Report of The ACM and IEEE-Computer Society Joint Task Force on Computing Curricula, Final Report, December 2001.

[Roberts 1999] Roberts E. Conserving the Seed Corn: Reflections on the Academic Hiring Crisis, SIGCSE Bulletin, (31)4:4-9, December 1999.

[Royce 1970] Royce W.W. Managing the Development of Large Software Systems: Concepts and Techniques, Proceedings of WESCON, August 1970.

[SAC 1967] President's Science Advisory Commission, Computers in Higher Education. Washington DC: The White House, February 1967.

[Saiedian 2002] Hossein S., Bagert D.J. and Nancy R. Mead Software Engineering Programs: Dispelling the Myths and Misconceptions, IEEE Software, V. 19, № 5, September/October, PP. 35-41, 2002.

[Shaw 1985] Shaw M. The Carnegie-Mellon curriculum for undergraduate computer science. New York: Springer-Verlag, 1985.

[Shaw 1990] Shaw M. Prospects for an Engineering Discipline of Software. IEEE Software, 7, 6, November 1990, PP. 15-24/

[Shaw 1991] Shaw M. and Tomayko J.E. Models for undergraduate courses in software engineering. Pittsburgh: Software Engineering Institute, Carnegie Mellon University, January 1991.

[Shaw 1992] Shaw M. We can teach software better. Computing Research News 4(4):2-12, September 1992.

[Shaw 2002] Shaw M. What makes good research in software engineering? International Journal on Software Tools for Technology Transfer, V. 4, DOI 10.1007/s10009-002-0083-4, June 2002/

[Shaw 2001] Shaw M. The Coming-of-Age of Software Architecture Research. Proceedings of the 23rd International Conference on Software Engineering, Toronto, PP. 656-664a, Canada, IEEE Computer Society, 2001.

[SIGCHI 1992] Special Interest Group on Computer-Human Interaction, ACM SIGCHI Curricula for Human-Computer Interaction, New York: Association for Computing Machinery, 1992.

[Sobel 2002] Sobel A.E.K. and Clarkson M. Formal Methods Application: An Empirical Tale of Software Development, IEEE Transactions on Software Engineering, V. 28. № 3, March 2002.

[Sobel 2001] Sobel A.E.K. Emphasizing Mathematical Analysis in a Software Engineering Curriculum, IEEE Transaction on Education, V. 44, № 2, CD-ROM, May 2001.

[Sobel 2000] Sobel A.E.K. Empirical Results of a Software Engineering Curriculum Incorporating Formal Methods, Proceedings of SIGCSE, 157-161, March 2000.

[Taipale, 1997] Taipale M. (Ed.) Proceedings of International Symposium on Software Engineering in Universities, ISSEU'97, Rovaniemi, Finland, March 1997.

[Thayer 1993] Thayer R.H. and McGettrick A. (Eds.), Software Engineering – a European Perspective, IEEE Computer Society Press, Los Alamitos, CA 1993.

[Thompson 2002] Thompson J.B. and Edwards H.M. Preliminary Report on the CSEET 2002 Workshop. Developing the Software Engineering Volume of Computing Curriculum 2001. Forum for Advancing Software Engineering Education (FASE), V. 12 № 3 (Issue 146), March 15, 2002.

[Thompson 2003] Thompson J.B. and Edwards H.M. Report on the 2nd International Summit on Software Engineering Education, ACM SIGSOFT Software Engineering Notes, V. 28, Issue 4 (July) PP. 21-26, 2003.

[Thompson 2004] Thompson J.B., Edwards H.M. and Lethbridge T.C. Post-Summit Proceedings International Summit on Software Engineering Education (SSEE), held on May 21, 2002 and co-located with the 24th IEEE-CS/ACM International Conference on Software Engineering (ICSE2002), in Orlando, Florida, University of Sunderland Press, Sunderland, UK, ISBN: 1-873757-34-4 (soft cover), 1-873757-89-1(CD), 2004.

[Tomayko 1999] Tomayko J. E. Forging a discipline: An outline history of software engineering education, Annals of Software Engineering, V. 6, № 1-4, PP. 3-18, April 1999.

[Tremblay 2000] Tremblay G. Formal Methods: Mathematics, Computer Science, or Software Engineering? IEEE Transactions on Education, V. 43, № 4, PP. 377-382, November 2000.

[Tucker 1991] Tucker A.B. et al. Computing Curricula '91, Association for Computing Machinery and the IEEE Computer Society, 1991.

[Umphress 2002] Umphress D.A., Hendrix T.D. and Cross J.H. Software Process in the Classroom: The Capstone Project Experience, IEEE Software, V. 19 , № 5, September/October, PP. 78-85, 2002.

[Walker 1996] Walker H.M. and Schneider G.M. A Revised Model Curriculum for a Liberal Arts Degree in Computer Science, Communications of the ACM, 39(12):85-95, December 1996.

[Zadeh 1968] Zade L.A. Computer Science as a Discipline, Journal of Engineering Education, 58(8):913-916, April 1968.

Приложение A. Подробное описание предлагаемых курсов

В этом приложении мы приводим подробное описание курсов, упоминавшихся в главе 6. Некоторые из этих курсов заимствованы из тома CCCS, тогда как другие курсы являются новыми и вводятся в данном томе впервые. Для новых курсов приводится следующая информация: полное описание курса, список предварительных требований к слушателям, задачи обучения и перечисление предполагаемого покрытия областей SEEK (см. главу 4) данным курсом. В некоторых случаях предложены также учебные модули, лабораторные работы и упражнения, а также приводятся прочие педагогические рекомендации. Для курсов из CCCS мы ограничиваемся только информацией о покрытии SEEK.

В большинстве случаев покрытие SEEK значительно меньше 40 лекционных часов, используемых в качестве стандарта для «полного» курса. Это дает возможность учебным заведениям и преподавателям подстраивать курсы под свои нужды, внося дополнительный материал или раскрывая глубже отдельные темы.

Вводные курсы CCCS

Поскольку эти курсы взяты непосредственно из тома CCCS, то читателю рекомендуется обращаться за дополнительными подробностями к первоисточнику [ACM 2001]. Отметим, что вместо приводимых ниже курсов могут использоваться и другие курсы из CCCS.

$CS101_I$. Основы программирования

Данный курс заимствован из тома по информатике (CCCS).

Описание курса:

Курс раскрывает основные понятия процедурного программирования. Темы включают типы данных, структуры управления, функции, массивы, файлы и механизмы запуска, тестирования и отладки. Курс также содержит введение в исторический и социальный контекст информатики и обзор информатики как научной дисциплины.

Требования к слушателям: Не требуется никакого предварительного опыта в области программирования или информатики. Студенты должны иметь достаточный объем математических знаний для решения простых линейных уравнений и уметь пользоваться математической нотацией и формализмами.

Программа курса:

- Компьютерные приложения: обработка текстов, электронные таблицы, файлы и каталоги.
- Базовые конструкции программирования: синтаксис и семантика языков высокого уровня; переменные, типы, выражения и присваивания; простейший ввод/вывод; условные предложения и итеративные конструкции; функции и передача параметров; структурная декомпозиция.

- Алгоритмы и решение задач: стратегии решения задач, роль алгоритмов в решении задач, стратегии реализации алгоритмов, стратегии отладки, понятие алгоритма, свойства алгоритмов.
- Базовые структуры данных: примитивные типы; массивы; структуры; строки и операции над строками.
- Представление данных в памяти компьютера: биты, байты, слова; представление числовых данных и системы счисления; представление символьных данных.
- Обзор операционных систем: роль и задачи операционных систем; простое управление файлами.
- Введение в распределенные вычисления: предпосылки возникновения и история сетей и Интернета; демонстрация и использование таких сетевых приложений, как электронная почта, Telnet и FTP.
- Человеко-машинное взаимодействие: введение в вопросы проектирования.
- Методология разработки программного обеспечения: основные понятия и принципы проектирования; структурная декомпозиция; стратегии тестирования и отладки; разработка сценариев тестирования (test cases); среды разработки; инструменты тестирования и отладки.
- Социальный контекст компьютинга: история компьютинга и компьютеров; эволюция идей и компьютеров; социальный эффект компьютеров и Интернета; профессионализм, кодекс этики и ответственное поведение; авторские права, интеллектуальная собственность и компьютерное пиратство.

Общее покрытие SEEK: 39 часов

CMP.cf (30 осн. часов из 140) – Основы информатики
- CMP.cf.1 (13 осн. часов из 39) – Основы программирования
- CMP.cf.2 (3 осн. часа из 31) – Алгоритмы, структуры и представление данных
- CMP.cf.3 (2 осн. часа из 5) – Методы решения задач
- CMP.cf.6 (1 осн. час из 1) – Базовые концепции систем
- CMP.cf.7 (1 осн. час из 1) – Человеческий фактор – пользователи
- CMP.cf.8 (1 осн. час из 1) – Человеческий фактор – разработчики
- CMP.cf.9 (7 осн. часов из 12) – Основы языков программирования
- CMP.cf.10 (1 осн. час из 10) – Основы операционных систем (ключевые концепции из CCCS)
- CMP.cf.12 (1 осн. час из 5) – Основы сетевых коммуникаций

CMP.tl (1 осн. час из 4) – Средства разработки

PRF.pr (4 осн. час из 20) – Профессионализм
- PRF.pr.2 – Кодексы этики и профессионального поведения
- PRF.pr.3 – Социальные, юридические, исторические и профессиональные вопросы
- PRF.pr.6 – Экономическое влияние программного обеспечения

MAA.rfd (1 осн. час из 3) – Основы управления требованиями

DES.con (1 осн. час из 3) – Концепции проектирования программного обеспечения
DES.con.1 – Определение проектирования
VAV.rev (1 осн. час из 6) – Рецензии кода
VAV.rev.1 – Проверка за столом
VAV.tst (1 осн. час из 21) – Тестирование
VAV.tst.1 – Модульное тестирование

$CS102_I$. Объектно-ориентированная парадигма

Данный курс заимствован из тома по информатике (CCCS).

Описание курса:

Данный курс знакомит студентов, обладающих представлением о процедурной парадигме, с понятиями объектно-ориентированного программирования. Курс начинается с обзора управляющих структур и типов данных с акцентом на структурные типы данных и работу с массивами. Затем вводится объектно-ориентированная парадигма программирования, с упором на определение и использование классов, а также на основы объектно-ориентированного проектирования. Другие темы курса включают в себя обзор принципов языков программирования, простой анализ алгоритмов, базовые методы поиска и сортировки и введение в вопросы программной инженерии.

Требования к слушателям: $CS101_I$.

Программа курса:

- Обзор управляющих структур, функций и примитивных типов данных.
- Объектно-ориентированное программирование: объектно-ориентированное проектирование, инкапсуляция и скрытие информации; разделение интерфейса и реализации; классы, наследники и наследование; полиморфизм; иерархии классов.
- Основные вычислительные алгоритмы: алгоритмы поиска и сортировки (линейный и дихотомический поиск, сортировка вставкой и выбором наименьшего элемента).
- Основы программирования, основанного на событиях.
- Введение в компьютерную графику: использование простых графических API.
- Обзор языков программирования: история языков программирования; краткий обзор парадигм программирования.
- Виртуальные машины: понятие виртуальной машины; иерархия виртуальных машин; промежуточные языки.
- Введение в теорию трансляции: сравнение интерпретаторов и компиляторов; стадии трансляции; машинно-зависимая и машинно-независимая части транслятора.
- Введение в СУБД: история и причины возникновения систем баз данных, использование языков запросов базы данных.

- Эволюция программ: сопровождение программ, характеристики удобного для сопровождения программного обеспечения, реинжиниринг, унаследованные системы, повторное использование программного обеспечения.

Общее покрытие SEEK: 36 часов

CMP.cf (30 осн. часов из 140) – Основы информатики
CMP.cf.1 (13 осн. часов из 39) – Основы программирования
CMP.cf.2 (3 осн. часа из 31) – Алгоритмы, структуры и представление данных
CMP.cf.3 (3 осн. часа из 5) – Методы решения задач
CMP.cf.4 (3 осн. часа из 5) – Использование и поддержка абстракции
CMP.cf.5 (2 осн. часа из 20) – Архитектура ЭВМ
CMP.cf.9 (5 осн. часов из 12) – Основы языков программирования
CMP.cf.11 (1 осн. час из 10) – Основы баз данных
CMP.ct (1 осн. час из 20) – Технологии разработки
DES.con.4 – Принципы проектирования
DES.hci (3 осн. часа из 12) – Проектирование человеко-машинного интерфейса
DES.hci.1 – Общие принципы проектирования человеко-машинного интерфейса
VAV.fnd (1 осн. час из 5) – Терминология и основы верификации и аттестации программного обеспечения
VAV.fnd.1 – Задачи и ограничения верификации и аттестации
EVO.pro (1 осн. час из 6) – Процессы эволюции
EVO.pro.1 – Основные концепции эволюции и сопровождения

CS103. Алгоритмы и структуры данных

Данный курс заимствован из тома по информатике (CCCS).

Описание курса:

Данный курс является логическим продолжением курсов $CS101_I$-102_I и знакомит студентов с базовыми концепциями структур данных и связанных с ними алгоритмов. Темы курса включают рекурсию, философию объектно-ориентированного программирования, базовые структуры данных (включая стеки, очереди, связанные списки, хэш-таблицы, деревья и графы), основы анализа алгоритмов и введение в принципы трансляции.

Требования к слушателям: $CS102_I$; желательно также знание дискретной математики в объеме курса CS105.

Программа курса:

- Обзор элементарных понятий программирования.
- Базовые структуры данных: стеки, очереди, связанные списки, хэш-таблицы, деревья, графы.
- Объектно-ориентированное программирование: объектно-ориентированное проектирование, инкапсуляция и скрытие информации, классы,

разделение интерфейса и реализации, иерархии классов, наследование, полиморфизм.

- Основные вычислительные алгоритмы: алгоритмы сортировки со сложностью O(NlogN), хэш-таблицы и алгоритмы избежания коллизий, двоичные деревья поиска, представления графов, обходы в глубину и в ширину.
- Рекурсия: понятие рекурсии, рекурсивные математические функции, простые рекурсивные процедуры, стратегия «разделяй и властвуй», рекурсивный перебор с возвратами, реализация рекурсии.
- Базовый анализ алгоритмов: асимптотический анализ максимальной и средней сложности; установление различий между лучшим, средним и худшим случаями; нотации «O-большое» и «o-маленькое», «омега» и «тета»; стандартные классы сложности; эмпирические измерения производительности; затраты по времени и объему памяти; использование рекуррентных соотношений для анализа рекурсивных алгоритмов.
- Алгоритмические стратегии: методы «грубой силы»; «жадные» алгоритмы; «разделяй и властвуй»; алгоритмы с возвратами; метод ветвей и границ; эвристики; сопоставление с образцом; алгоритмы обработки строк и текстов; алгоритмы численной аппроксимации.
- Обзор языков программирования: парадигмы программирования.
- Программная инженерия: аттестация программного обеспечения; основы тестирования, включая создание плана тестирования и генерации тестовых сценариев; объектно-ориентированное тестирование.

Общее покрытие SEEK: 31 час

CMP.cf (30 осн. часов из 140) – Основы информатики
CMP.cf.1 (13 осн. часов из 39) – Основы программирования
CMP.cf.2 (15 осн. часов из 31) – Алгоритмы, структуры и представление данных
CMP.cf.4 (2 осн. часа из 5) – Использование и поддержка абстракции
CMP.cf.9 – Основы языков программирования
VAV.tst (1 осн. час из 21) – Тестирование
VAV.tst.2 – Обработка исключений

Промежуточные фундаментальные курсы по информатике

В данном разделе приведен примерный набор курсов из SE2004, которые могут использоваться для преподавания обязательных разделов SEEK. Можно использовать и другие комбинации курсов из CCCS, а также создавать новые курсы для покрытия данного материала. При использовании приведенной ниже последовательности из трех курсов студенты изучат существенно больший объем материала, чем это требуется для покрытия основ SEEK. Мы полагаем, что во многих программах обучения программной инженерии будет преподаваться такой же объем знаний по информатике, как в перечисленных ниже курсах, или даже более.

CS220. Архитектура ЭВМ

Данный курс заимствован из тома по информатике (CCCS).

Описание курса:

Данный курс знакомит студентов с организацией и архитектурой компьютерных систем, начиная со стандартной модели фон Неймана и заканчивая новейшими понятиями в архитектуре ЭВМ.

Требования к слушателям: введение в информатику (любая реализация курса CS103 или CS112), дискретные структуры (CS106 или CS115).

Программа курса:

- Цифровая логика: основные составные блоки (логические вентили, триггеры, счетчики, регистры, программируемая логическая матрица); логические выражения, минимизация, конъюнктивные и дизъюнктивные формы; нотация пересылки регистров; физические аспекты (задержки вентилей, нагрузочные способности по входу и по выходу (fan-in, fan-out)).
- Представление данных: биты, байты, слова; представление числовых данных и системы счисления; системы с фиксированной и с плавающей запятой; знаковые представления и представления в дополнительном коде; представление нечисловых данных (коды символов, графические данные); представление записей и массивов.
- Организация машины на уровне ассемблера: основы организации фоннеймановской машины; управляющее устройство; выборка, дешифрация и выполнение команд; системы команд и типы команд (обработка данных, управление, ввод/вывод); программирование на языке ассемблера; форматы инструкций; режимы адресации; механизмы вызова подпрограмм и возврата из них; ввод/вывод и прерывания.
- Организация памяти: системы и технологии хранения; кодирование, сжатие и целостность данных; иерархия памяти; организация и принципы работы основной памяти; задержка (latency), длительность такта (cycle time), пропускная способность (bandwidth) и чередование (interleaving); кэш-память (преобразование адресов, размер блока, политика замещения и хранения); виртуальная память (таблица страниц, TLB – буфер быстрого преобразования адреса); обработка ошибок доступа к памяти и надежность.
- Организация взаимодействия устройств: основы ввода/вывода; установление связи, буферизация, программируемый ввод/вывод, ввод/вывод по прерыванию; структура прерываний: векторная и приоритетная; подтверждение прерываний; внешние накопители, физическая организация и диски; шины: протоколы, арбитраж, прямой доступ к памяти; знакомство с сетями; поддержка мультимедиа; RAID-архитектуры.
- Функциональная организация устройств: реализация схем с простой передачей данных; управляющее устройство; сравнение аппаратной и мик-

ропрограммной реализаций; конвейер инструкций; введение в параллелизм уровня машинных команд.

- Многопроцессорные и альтернативные архитектуры: знакомство с SIMD, MIMD, VLIW, EPIC; систолическая архитектура; взаимодействие сетей; архитектуры с общей памятью; обеспечение соответствия кэша и памяти; модели и целостность памяти.
- Увеличение производительности устройств: RISC-архитектура; предсказывание переходов; «водопровод» (prefetching); масштабирование.
- Современные архитектуры: мобильные устройства, встроенные системы, направления развития архитектуры процессоров.

Общее покрытие SEEK: 15 часов

CMP.cf (15 осн. часов из 140) – Основы информатики

CMP.cf.5 (15 осн. часов из 20) – Архитектура ЭВМ

CS226. Операционные системы и сети

Данный курс заимствован из тома по информатике (CCCS).

Описание курса:

Данный курс знакомит студентов с основами операционных систем, а также с основами сетей и телекоммуникаций.

Требования к слушателям: введение в информатику (любая реализация курса CS103 или CS112), дискретные структуры (CS106 или CS115).

Программа курса:

- Знакомство с программированием событийно-управляемых систем.
- Использование программных интерфейсов приложения (API): программирование с использованием API; программы просмотра классов и другие подобные инструменты; программирование с помощью примеров; отладка программы, использующей API.
- Обзор операционных систем: роль и задачи операционных систем; история развития операционных систем; функциональность типичной операционной системы.
- Основные принципы работы операционных систем: методы структуризации; абстракции, процессы и ресурсы; понятие программных интерфейсов приложений; организация устройств; прерывания; понятия режимов работы пользователя/супервизора и защиты.
- Введение в параллелизм: принципы синхронизации; проблема взаимного исключения и некоторые решения; избегание блокировок.
- Параллелизм: состояния и диаграммы состояний; структуры; диспетчеризация и переключение контекстов; роль прерываний; параллельное исполнение; проблема взаимного исключения и некоторые решения; блокировки; модели и механизмы; проблемы поставщика/потребителя и синхронизация.

- Планирование и диспетчеризация: вытесняющее и невытесняющее планирование; планировщики и политики их работы; процессы и нити; учет предельных сроков и реального времени.
- Управление памятью: обзор физической памяти и управляющей аппаратуры; оверлеи, подкачка и разделы; страничная организация памяти и сегментация; стратегии подкачки и выгрузки страниц; рабочие множества и пробуксовка; кэширование.
- Введение в распределенные алгоритмы: консенсус и выборы; отказоустойчивость
- Введение в распределенное программирование: причины возникновения и история сетевого программирования и Интернета; сетевые архитектуры; круг специализаций в сетевом программировании.
- Введение в телекоммуникации и сети: сетевые архитектуры; вопросы, связанные с распределенными вычислениями; простые сетевые протоколы; API для выполнения сетевых операций.
- Знакомство с WWW: web-технологии; характеристики web-серверов; природа связи клиент-сервер; web-протоколы; программные средства для создания и управления web-сайтом.
- Сетевая безопасность: основы криптографии; алгоритмы с секретным ключом; алгоритмы с открытым ключом; протоколы аутентификации; цифровые подписи; примеры.

Общее покрытие SEEK: 16 часов

CMP.cf. Основы информатики

CMP.cf.2 (3 осн. часа из 31) – Алгоритмы, структуры и представление данных

CMP.cf.10 (9 осн. часов из 10) – Основы операционных систем (ключевые понятия из CCCS)

CMP.cf.12 (4 осн. часа из 5) – Основы сетевых коммуникаций

CS270T. Базы данных

Данный курс заимствован из тома по информатике (CCCS).

Описание курса:

Данный курс знакомит студентов с понятиями и методами работы баз данных.

Требования к слушателям: введение в информатику (любая реализация курса CS103 или CS112), дискретные структуры (CS106 или CS115).

Программа курса:

- Информационные модели и системы: история и причины возникновения информационных систем; хранение и поиск информации; приложения, управляющие информацией; сбор и представление информации; анализ и индексирование; поиск, получение, связывание и навигация; конфиденциальность, целостность, безопасность и сохранение информации; масштабируемость, производительность и эффективность.

- Системы баз данных: история и причины возникновения систем баз данных; компоненты базы данных; функциональность СУБД; архитектура базы данных и независимость данных.
- Моделирование данных: моделирование данных; концептуальные модели; объектно-ориентированная модель; реляционная модель данных.
- Реляционные базы данных: отображение концептуальной схемы в реляционную схему; целостность сущностей-объектов и ссылочная целостность; реляционная алгебра и реляционное исчисление.
- Языки запросов к базам данных: обзор языков баз данных; SQL; оптимизация запросов; окружения 4-го поколения; встраивание непроцедурных запросов в процедурный язык; введение в объектно-ориентированный язык запросов (OQL).
- Структура реляционных баз данных: структура баз данных; функциональные зависимости; нормальные формы; многозначные зависимости; зависимость соединения; теория представления данных.
- Обработка транзакций: транзакции; неудачи и восстановление; управление параллелизмом.
- Распределенные базы данных: распределенное хранение данных; обработка распределенных запросов; распределенная модель транзакций; управление параллелизмом; однородные и гетерогенные решения; клиент-серверная архитектура.
- Физическое устройство баз данных: структура файлов; индексированные файлы; файлы с хэшированным доступом; файлы сигнатур; Б-деревья; файлы с плотным индексом; файлы с записями переменной длины; производительность базы данных и настройка.

Общее покрытие SEEK: 13 часов

CMP.cf (11 осн. часов из 140) – Основы информатики

CMP.cf.2 (2 осн. часа из 31) – Алгоритмы, структуры и представление данных

CMP.cf.11 (9 осн. часов из 10) – Основы баз данных

MAA.md (2 осн. часа из 19) – Основы моделирования

Фундаментальные курсы по математике

CS105. Дискретные структуры I

Данный курс заимствован из тома по информатике (CCCS).

Описание курса:

Данный курс знакомит студентов с основами дискретной математики и ее применениями в информатике. Цель данного курса – создать надежный теоретический фундамент для последующих курсов. Здесь обсуждаются функции, отношения, множества, простейшие методы доказательства, булева алгебра, логика высказываний, цифровая логика, элементарная теория чисел и основы комбинаторики.

Требования к слушателям: математическая подготовка, позволяющая заниматься математикой на университетском уровне.

Программа курса:

- Знакомство с логикой и доказательствами; прямые доказательства; доказательства от противного; математическая индукция.
- Основные объекты: функции (сюръекции, инъекции, инверсии, композиция отображений); отношения (рефлексивные, симметричные, транзитивные, эквивалентность); множества (диаграммы Венна, дополнения, декартовы произведения, степенные множества); принцип Дирихле; мощность и счетность.
- Булева алгебра: логические значения; стандартные операции над логическими значениями; законы де Моргана.
- Логика высказываний: логические связки; таблицы истинности; нормальные формы (конъюнктивная и дизъюнктивная); общезначимость.
- Цифровая логика: логические вентили; триггеры; минимизация цепей.
- Элементарная теория чисел: разложимость на множители; свойства простых чисел; наибольший общий делитель и наименьшее общее частное; алгоритм Эвклида; арифметические операции над остатками; китайская теорема об остатках.
- Основы комбинаторики: комбинаторные объекты; принцип Дирихле; перестановки и подстановки; биномиальные коэффициенты.

Общее покрытие SEEK: 24 часа

CMP.cf (3 осн. часа из 140) – Основы информатики
CMP.cf.5 (3 осн. часа из 20) – Архитектура ЭВМ
FND.mf (21 осн. час из 56) – Основы математики
FND.mf.1 (6 осн. часов из 6) – Функции, отношения и множества
FND.mf.2 (5 осн. часов из 9) – Основы логики
FND.mf.3 (4 осн. часа из 9) – Методы доказательства
FND.mf.4 (6 осн. часов из 6) – Основы вычислений
FND.mf.10 – Теория чисел

CS106. Дискретные структуры II

Данный курс заимствован из тома по информатике (CCCS).

Описание курса:

Данный курс продолжает изложение дискретной математики, начатое в курсе CS105. Темы данного курса включают в себя логику предикатов, рекуррентные соотношения, графы, деревья, матрицы, вычислительную сложность, элементарную вычислимость и дискретную вероятность.

Требования к слушателям: CS105.

Программа курса:

- Обзор предыдущего курса.

- Логика предикатов: применение квантора всеобщности и квантора существования; правила введения конъюнкции и дизъюнкции; ограничения логики предикатов.
- Рекуррентные соотношения: основные формулы; элементарные методы решения.
- Графы и деревья: основные определения; простейшие алгоритмы; стратегии обхода; методы доказательств; остовные деревья; приложения.
- Матрицы: основные свойства; приложения.
- Вычислительная сложность: порядковый анализ; стандартные классы сложности.
- Элементарная вычислимость: счетность и несчетность; диагональный метод доказательства несчетности континуума; определения классов P и NP; простая демонстрация проблемы останова.
- Дискретная вероятность: конечные вероятностные пространства; условная вероятность, вероятности независимых событий, формулы Байеса; случайные события; случайные целочисленные величины; математическое ожидание.

Общее покрытие SEEK: 27 часов

CMP.cf (5 осн. часов из 140) – Основы информатики
CMP.cf.2 (5 осн. часов из 31) – Алгоритмы, структуры и представление данных
FND.mf (19 осн. часов из 56) – Основы математики
FND.mf.2 (4 осн. часа из 9) – Основы логики
FND.mf.3 (5 осн. часов из 9) – Методы доказательства
FND.mf.4 (0 осн. часов из 6) – Основы вычислений
FND.mf.5 (4 осн. часа из 5) – Графы и деревья
FND.mf.6 (6 осн. часов из 9) – Дискретная теория вероятности
MAA.md (3 осн. часа из 19) – Основы моделирования

MA271. Статистические и эмпирические методы компьютинга

Данный новый курс вводится как часть тома по программной инженерии, несмотря на то, что покрываемые в нем темы сами по себе не входят в предметную область программной инженерии. Потребность в этом курсе обусловливается прежде всего желанием обучать теории вероятностей и статистике как прикладным наукам, актуальным для студентов, изучающих программную инженерию. Конечно, возможно заменить этот курс более общими статистическими курсами, однако опыт многих преподавателей говорит, что студенты быстро забывают приобретенные знания, если не воспринимают их как значимые для своей последующей карьеры. Мы надеемся, что данный курс в какой-то степени исправит имеющееся положение вещей.

Описание курса:

Применение принципов дискретной вероятности в компьютинге. Основы описательной статистики. Распределения, включая нормальное (Гауссово), бино-

миальное и Пуассона. Концепция метода наименьших квадратов, корреляция и регрессия. Статистические тесты, наиболее актуальные для программной инженерии: проверка по критерию Стьюдента, дисперсионный анализ и критерий ?2. Планирование экспериментов и проверка гипотез. Статистический анализ данных из различных источников. Применение статистики для анализа производительности и надежности системы, проектирования удобства использования, оценки стоимости и оценки процесса управления.

Задачи обучения:

Студенты, успешно прослушавшие данный курс, должны уметь:

- принимать проектные и управленческие решения, исходя из хорошего понимания теории вероятностей и статистики;
- планировать и осуществлять эксперименты для оценки гипотез о качестве и процессе разработки программного обеспечения;
- анализировать данные из различных источников;
- понимать важность эмпирических методов в программной инженерии.

Примеры лабораторных заданий:

- Создание электронных таблиц на основе данных, полученных в результате различных экспериментов, с дальнейшим использованием встроенных статистических функций для проверки гипотез.
- Использование статистических пакетов, например, SAS или SPSS.

Дополнительные рекомендации по обучению:

- Некоторые преподаватели любят выводить статистические методы от первооснов и тратят большую часть времени курса по статистике на рассмотрение и доказательство теорем. Мы полагаем, что материал, который преподается таким образом, имеет тенденцию быстро забываться всеми студентами, кроме особо склонных к математике, и потому такая подача материала зачастую является потерей времени. Вместо этого мы предлагаем преподавать методы статистики как рецепты из «кулинарной книги», приведя, однако, достаточное математическое обоснование, чтобы студенты могли последовательно расширять свои знания. Используя такой подход, студенты могут в последующих (факультативных) курсах подробнее изучить математические основы статистики и/или широкий спектр методов анализа данных.
- Рекомендуется использовать в курсе электронные таблицы в дополнение к статистическим пакетам, так как все IT-компании используют электронные таблицы, но не все из них могут или хотят приобрести более мощные, сложные и дорогие статистические пакеты. Студенты с большей вероятностью смогут применять в дальнейшем статистику, если они знают, как это делать с помощью электронных таблиц.
- Данный курс может быть связан с другими курсами по программной инженерии, которые читаются параллельно, например с SE212, SE321 или SE323. Вне зависимости от того, читаются ли эти курсы одновременно,

необходимо сопровождать материал упражнениями для закрепления изучаемого материала.

Общее покрытие SEEK: 18 часов

FND.mf (3 осн. часа из 56) – Основы математики

FND.mf.6 (3 осн. часа из 9) – Дискретная теория вероятности

FND.ef (15 осн. часов из 23) – Инженерные основы программного обеспечения

FND.ef.1 – Эмпирические методы и экспериментальные методы

FND.ef.2 – Статистический анализ

Нетехнические обязательные курсы

Общее покрытие SEEK следующей серией курсов гораздо меньше 40 часов, поэтому у образовательных учреждений есть свобода для разработки этих курсов самостоятельно, с учетом их потребностей.

NT272. Инженерная экономика

Подобные курсы широко преподаются на инженерных факультетах, в частности, в Северной Америке. Приведенный ниже курс может использоваться в инженерной программе для любого типа инженерии. Он может более глубоко учитывать потребности программной инженерии.

Описание курса:

Рамки инженерной экономики, мезоэкономика (mesoeconomics); спрос, предложение и выработка; анализ затрат и результатов, анализ уровня безубыточности, прибыль на инвестированный капитал (ROI), анализ альтернатив, временная ценность денег, финансовое управление: экономический анализ, расчет рисков.

Задачи обучения:

Студенты, успешно прослушавшие данный курс, должны уметь:

- анализировать спрос и предложение на продукт;
- осуществлять простой анализ уровня безубыточности;
- осуществлять простой анализ затрат и результатов;
- анализировать экономический эффект альтернативных решений по инвестированию, маркетингу и проектированию, принимая во внимание временную ценность денег и потенциальный риск.

Общее покрытие SEEK: 13 часов

FND.ef (2 осн. часа из 23) – Инженерные основы программного обеспечения

FND.ef.5 – Инженерное проектирование

FND.ec (10 осн. часов из 10) – Инженерная экономика программного обеспечения

MGT.pp (1 осн. час из 6) – Планирование проекта

NT181. Групповая динамика и коммуникации

Описание курса:

Основы устной, письменной и графической коммуникации для программных инженеров. Принципы написания документации, типы документов, включая презентации. Уместное использование таблиц, графиков и ссылок. Как быть убедительным и как ясно раскрыть суть своего решения или заключения. Основы эффективной работы с коллегами, знакомство с мотивацией людей, концепции групповой динамики. Принципы эффективной устной коммуникации как в межличностном общении, так и при проведении презентации для группы. Стратегии выслушивания, убеждения и ведения переговоров.

Задачи обучения:

Студенты, успешно прослушавшие данный курс, должны уметь:

- создавать ясную, лаконичную и точную техническую документацию, следуя четким стандартам по форматированию и по включению подходящих таблиц, рисунков и ссылок;
- рецензировать письменную техническую документацию с целью обнаружения различного рода проблем;
- разрабатывать и проводить формальные презентации хорошего качества;
- договариваться о базовых соглашениях с партнерами;
- участвовать в таком взаимодействии с другими людьми, при котором люди смогут понять, выслушать и оценить позицию друг друга, даже если они не согласны, а также смогут донести до других свою позицию.

Дополнительные рекомендации по обучению:

- Некоторые студенты не могут ясно излагать свои мысли на бумаге, поэтому одной из задач данного курса является помощь студентам в улучшении этих навыков. Однако предполагается, что работа по повторению грамматики не является частью основного курса, поскольку это будет потерей времени для тех студентов, которым это не требуется. Помощь по улучшению грамматики следует отдельно предоставлять тем студентам, которые в ней нуждаются. Следует очень критично оценивать стиль письма каждого из студентов, не следует заканчивать обучение до тех пор, пока студент не научится хорошо писать документацию.
- Преподавателю следует дать задание студенту написать несколько документов умеренного размера, делая ударение на ясности, полезности и качестве изложения. Предполагается избегать документов сложных форматов.
- Студентам можно дать задание изложить требования, для того чтобы описать то, как что-то работает, или описать, как что-то сделать. Эти темы наилучшим образом подготовят студентов для написания документации тех типов, которые им понадобятся как будущим инженерам по программному обеспечению. Назначенные темы должны быть интересными

для студентов, с целью их лучшей мотивации. Например, можно предложить описать игру.

Общее покрытие SEEK: 11 часов

PRF.psy (3 осн. часа из 5) – Групповая динамика и психология

PRF.com (8 осн. часов из 10) – Навыки коммуникации

MAA.rsd.1 – Основы документирования требований

NT291. Профессиональная практика в программной инженерии

Описание курса:

История компьютинга и программной инженерии. Принципы профессиональной деятельности и этики программной инженерии. Общественные обязательства и обязательства по охране окружающей среды. Роль профессиональных организаций. Защита интеллектуальной собственности и другое законодательство, значимое для деятельности по программной инженерии.

Задачи обучения:

Студенты, успешно прослушавшие данный курс, должны уметь:

- принимать этические решения, при столкновении с этическими дилеммами, ссылаясь как на общие этические принципы, так и на этические кодексы инженерии, компьютинга и программной инженерии;
- уделять внимание безопасности, защищенности и правам человека в инженерии и управлении принятием решений;
- знать основы истории инженерии, компьютинга и программной инженерии;
- разъяснять и применять законодательство, которое затрагивает программную инженерию, включая законодательство по авторскому праву, патентам и прочим аспектам интеллектуальной собственности;
- описывать эффект, который оказывают решения в программной инженерии на общество, экономику, социальную среду, их заказчиков, руководство, партнеров и на них самих;
- описать важность множества различных сообществ, значимых для программной инженерии на региональном уровне, в стране, а также на международной арене;
- понимать роль стандартов и определяющих стандарты совокупностей знаний в инженерии и программной инженерии;
- осознавать потребность в постоянном повышении своей квалификации как инженера, так и инженера по программному обеспечению.

Дополнительные рекомендации по обучению:

- Предлагается использовать приглашенных лекторов для прочтения некоторых частей этого курса. Например, это могут быть эксперты по этике, представители профессионального сообщества, эксперты по интеллектуальной собственности и т.п.

- Студентам следует задавать читать и обсуждать значимые для курса статьи из популярной, коммерческой и академической прессы.
- Студентам следует дискутировать по различным этическим вопросам.
- Следует следить за тем, чтобы были представлены обе точки зрения по обсуждаемым вопросам. В частности, мы полагаем, что должны быть представлены оба мнения за и против лицензирования инженера по программному обеспечению, поскольку уважаемые лидеры в этой профессии все еще придерживаются диаметрально противоположных взглядов на этот вопрос. Другой вопрос, обе стороны которого важно осветить, заключается в патентовании программного обеспечения. Мы думаем, что целиком приемлемо для преподавателя защищать свои «политические» мнения по этим вопросам, в то время как студенты смогут придерживаться мнения «другой стороны», не подвергаясь «репрессиям» за оппонирование точке зрения преподавателя.

Общее покрытие SEEK: 14 часов.

PRF.pr (13 осн. часов из 20) – Профессионализм
- PRF.pr.1 – Аккредитация, сертификация и лицензирование
- PRF.pr.2 – Кодекс этики и профессионального поведения
- PRF.pr.3 – Социальные, юридические, исторические и профессиональные вопросы
- PRF.pr.4 – Происхождение и значение профессиональных сообществ
- PRF.pr.5 – Происхождение и значение стандартов программной инженерии
- PRF.pr.6 – Экономическое влияние развития программного обеспечения

QUA.cc (1 осн. час из 2) – Концепции и культура качества программного обеспечения
- QUA.cc.2 – Общественная заинтересованность в качестве
- QUA.cc.3 – Стоимость и ущерб от плохого качества

Вводные курсы по SE+CS – первый год обучения

SE101. Программная инженерия и компьютинг

Данный курс является первым курсом по компьютингу, в котором акцент делается на программную инженерию. Он разработан для прочтения совместно с SE102 в качестве замены для курсов CS101 и CS102 из тома CCCS. Курсы по информатике учат основам программной инженерии, однако идея состоит в том, что этот курс начинается с материала по программной инженерии и подает этот материал в виде методичного решения задач программной инженерии для заказчиков.

Описание курса:

Обзор программной инженерии: системы, заказчики, пользователи и их требования. Общие принципы компьютинга: решение задач, абстракция, разделение системы на управляемые компоненты, повторное использование, простые интерфейсы. Концепции программирования: управляющие конструкции, выражения, использо-

вание API, простые данные, включая массивы и строки, классы и наследование. Концепции проектирования: оценивание альтернатив. Основы тестирования.

Требования к слушателям:

Законченное среднее образование с высокими оценками, внимательное отношение к деталям, развитое при изучении математики и точных наук.

Задачи обучения:

Студенты, успешно прослушавшие данный курс, должны уметь:

- разрабатывать простые описания требований;
- оценить преимущество альтернативных наборов требований и проектов для очень простых программ;
- писать небольшие программы на одном из языков программирования;
- систематически тестировать и отлаживать небольшие программы.

Дополнительные рекомендации по обучению:

Так как данный курс является первым курсом по компьютингу, то основной проблемой будет заинтересовать студентов программной инженерией до того, как узнают что-либо о программировании. Одним из хороших способов сделать это является изучение простых программ снаружи (черные ящики), рассмотрение их свойств и обсуждение возможности их улучшения. Однако это следует сделать с достаточной научной строгостью. В качестве одного из подходов к курсу возможно разбиение на два параллельных потока (другими словами, на два синхронизированных мини-курса). В одном потоке рассматриваются высокоуровневые вопросы программной инженерии, в то время как в другом происходит обучение программированию.

Общее покрытие SEEK: 35 часов

CMP.cf (19 осн. часов из 140) – Основы информатики
- CMP.cf.1 (9 осн. часов из 39) – Основы программирования
- CMP.cf.3 (2 осн. часа из 5) – Методы решения задач
- CMP.cf.4 (1 осн. час из 5) – Использование и поддержка абстракции
- CMP.cf.5 (2 осн. часа из 20) – Архитектура ЭВМ
- CMP.cf.6 (1 осн. час из 1) – Базовые концепции систем
- CMP.cf.7 (1 осн. час из 1) – Человеческий фактор – пользователи
- CMP.cf.8 (1 осн. час из 1) – Человеческий фактор – разработчики
- CMP.cf.9 (2 осн. часа из 12) – Основы языков программирования

CMP.ct (2 осн. часа из 20) – Технологии разработки

CMP.tl (1 осн. час из 4) – Средства разработки

FND.ef (2 осн. часа из 23) – Инженерные основы программного обеспечения
- FND.ef.3 – Измерение индивидуальной производительности
- FND.ef.4 – Разработка систем
- FND.ef.5 – Инженерное проектирование

PRF.pr (2 осн. часа из 20) – Профессионализм

MAA.tm (1 осн. час из 12) – Типы моделей

MAA.rfd (2 осн. часа из 3) – Основы управления требованиями

MAA.er (1 осн. час из 4) – Выявление требований

MAA.rsd (1 осн. час из 6) – Документирование и спецификация требований
DES.con (1 осн. час из 3) – Концепции проектирования программного обеспечения
DES.str (1 осн. час из 6) – Стратегии проектирования программного обеспечения
DES.dd (1 осн. час из 12) – Детальное проектирование
VAV.tst (1 осн. час из 21) – Тестирование

SE102. Программная инженерия и компьютинг II

Данный курс является продолжением курса SE101 для студентов, ориентированных на модель обучения «сначала программирование, затем программная инженерия».

Описание курса:

Требования, проектирование, реализация, рецензирование и тестирование простого программного обеспечения, которое взаимодействует с операционной системой, базами данных, сетью и использует графические пользовательские интерфейсы.

Использование простых структур данных, таких как стеки и очереди. Эффективное использование средств языков программирования. Проектирование и анализ простых алгоритмов с использованием рекурсии. Использование простых шаблонов проектирования, таких как делегирование. Создание простых UML-диаграмм классов, пакетов и компонентов. Работа с изменениями: принципы эволюции, управление изменениями требований, отчеты о проблемах и их отслеживание.

Требования к слушателям: SE101

Задачи обучения:

Студенты, успешно прослушавшие данный курс, должны уметь:

- разрабатывать ясные, лаконичные и достаточно формальные описания требований для расширения уже существующих систем, основываясь на насущных потребностях пользователей и других заинтересованных лиц;
- проектировать программное обеспечение таким образом, чтобы его легко изменять;
- проектировать простые алгоритмы с использованием рекурсии;
- анализировать базовые алгоритмы для определения их эффективности;
- создавать простые диаграммы для представления проекта программного обеспечения;
- коллективно разрабатывать программы среднего размера;
- разрабатывать простой графический интерфейс пользователя;
- проводить инспекции программ среднего размера.

Дополнительные рекомендации по обучению:

Так же как и в SE101, студентам следует регулярно напоминать о принципах программной инженерии.

Общее покрытие SEEK: 36 часов

CMP.cf (23 осн. часа из 140) – Основы информатики
CMP.cf.1 (12 осн. часов из 39) – Основы программирования
CMP.cf.3 (3 осн. часа из 5) – Методы решения задач
CMP.cf.4 (1 осн. час из 5) – Использование и поддержка абстракции
CMP.cf.9 (4 осн. часа из 12) – Основы языков программирования
CMP.cf.10 (1 осн. час из 10) – Основы операционных систем (ключевые понятия из CCCS)
CMP.cf.11 (1 осн. час из 10) – Основы баз данных
CMP.cf.12 (1 осн. час из 5) – Основы сетевых коммуникаций
PRF.pr (1 осн. час из 20) – Профессионализм
MAA.md (1 осн. час из 19) – Основы моделирования
MAA.rv (1 осн. час из 3) – Проверка требований
DES.str (1 осн. час из 6) – Стратегии проектирования программного обеспечения
DES.dd (1 осн. час из 12) – Детальное проектирование
DES.nst (1 осн. час из 3) – Нотации и средства проектирования
VAV.fnd (1 осн. час из 5) – Терминология и основы верификациии b аттестации программного обеспечения
VAV.rev (1 осн. час из 6) – Рецензии кода
VAV.tst (2 осн. часа из 21) – Тестирование
VAV.par (1 осн. час из 4) – Анализ проблем и создание отчетов
EVO.pro (1 осн. час из 6) – Процессы эволюции

Базовые курсы по программной инженерии

SE200. Программная инженерия и компьютинг III

Данный курс является третьим в группе для студентов, которые следуют последовательности SE101, SE102.

Описание курса:

Процесс разработки программного обеспечения, планирование и отслеживание выполнения работы. Анализ, архитектура и проектирование простых систем «клиент-сервер» с использованием UML и акцентом на диаграммах классов и состояний. Оценивание проектов. Реализация проектов с использованием подходящих структур данных, сред и API.

Требования к слушателям: SE102.

Задачи обучения:

Студенты, успешно прослушавшие данный курс, должны уметь:

- планировать разработку простых систем;
- измерять и отслеживать их прогресс во время разработки программного обеспечения;
- создавать хорошие UML-диаграммы классов и состояний;

- реализовывать системы значительной сложности.

Дополнительные рекомендации по обучению:

Данный курс предоставляет хорошую возможность начать вовлекать студентов в работу с системами умеренных размеров, что позволяет студентам развивать важные навыки чтения и понимания кода, написанного другими.

В отличие от SE201, этот курс должен соблюдать баланс между изучением программной инженерии и непрерывным изучением программирования и базовых компьютерных наук. Предполагается, что преподаются основы UML, но без попыток охватить весь язык.

Общее покрытие SEEK: 38 часов

CMP.cf (18 осн. часов из 140) – Основы информатики
- CMP.cf.1 (5 осн. часов из 39) – Основы программирования
- CMP.cf.2 (6 осн. часов из 31) – Алгоритмы, структуры и представление данных
- CMP.cf.4 (1 осн. час из 5) – Использование и поддержка абстракции
- CMP.cf.9 (6 осн. часов из 12) – Основы языков программирования

CMP.ct (3 осн. часа из 20) – Технологии разработки
FND.ef (1 осн. час из 23) – Инженерные основы программного обеспечения
PRF.pr (2 осн. часа из 20) – Профессионализм
MAA.md (1 осн. час из 19) – Основы моделирования
DES.con(2 осн. часа из 3) – Концепции проектирования программного обеспечения
DES.str (1 осн. час из 6) – Стратегии проектирования программного обеспечения
DES.ar (2 осн. часа из 9) – Архитектурное проектирование
DES.hci (4 осн. часа из 12) – Проектирование человеко-машинного интерфейса
DES.ev (1 осн. час из 3) – Оценка дизайна
VAV.fnd (1 осн. час из 5) – Терминология и основы верификации и аттестации программного обеспечения
VAV.rev (1 осн. час из 6) – Рецензии кода
PRO.imp (1 осн. час из 10) – Внедрение процессов
MGT.con (1 осн. час из 2) – Концепции менеджмента

SE201. Введение в программную инженерию II

Данный курс является первым курсом по программной инженерии, который прослушивается студентами после курсов CS101 и CS102.

Описание курса:

Принципы программной инженерии: требования, проектирование и тестирование. Обзор принципов объектно-ориентированного программирования. Объектно-ориентированный анализ с использованием UML. Среды и API. Введение в архитектуру «клиент-сервер». Анализ, проектирование и программирование

простых серверов и клиентов. Введение в технологию пользовательского интерфейса.

Требования к слушателям: CS102.

Задачи обучения:

Студенты, успешно прослушавшие данный курс, должны уметь:

- разрабатывать ясные, лаконичные и достаточно формальные описания требований для расширения уже существующих систем, основываясь на насущных потребностях пользователей и других заинтересованных лиц;
- использовать принципы и шаблоны проектирования для проектирования и реализации простых распределенных систем на основе повторного использования технологий;
- создавать диаграммы классов на UML, которые документируют модель предметной области и программную архитектуру;
- создавать диаграммы последовательности и состояний на UML, корректно моделирующие поведение системы;
- реализовывать простой графический пользовательский интерфейс системы;
- использовать простые методы измерения при разработке программного обеспечения;
- демонстрировать понимание многогранности программной инженерии.

Рекомендуемая последовательность преподаваемых модулей:

1. Программная инженерия и ее место среди инженерных дисциплин.
2. Обзор принципов объектно-ориентированного подхода.
3. Технологии повторного использования как фундамент программной инженерии: среды и API, введение в архитектуру «клиент-сервер».
4. Анализ требований.
5. Диаграммы классов на UML и объектно-ориентированный анализ, введение в формальное моделирование с использованием OCL.
6. Примеры построения диаграмм классов для различных предметных областей.
7. Шаблоны проектирования (абстрактное-конкретное (abstraction-occurrence), компоновщик (composite), игрок-роль (player-role), одиночка (singleton), наблюдатель (observer), делегирование (delegation), фасад (facade), адаптер (adapter)).
8. Варианты использования (use cases) и проектирование, ориентированное на пользователя.
9. Представление поведения программного обеспечения: диаграммы последовательности, состояний, деятельности.
10. Общие принципы разработки программного обеспечения: декомпозиция (decomposition), несвязанность (decoupling), зацепление (cohesion), повторное использование (reuse), возможность повторного использова-

ния (reusability), переносимость (portability), тестируемость (testability), гибкость (flexibility) и т.п.

11. Программная архитектура: распределенные архитектуры, «каналы-и-фильтры» (pipe-and-filter), «модель-вид-контроллер» (model-view-controller) и т.д.
12. Введение в тестирование и управление проектами.

Примеры лабораторных работ и заданий:

- Оценка эффективности различных простых проектных решений.
- Добавление новых возможностей в существующую систему.
- Тестирование системы для проверки соответствия тестовым сценариям.
- Построение графического интерфейса пользователя для приложения.
- Различные упражнения по построению моделей с использованием UML, в частности диаграмм классов и состояний.
- Разработка простого набора требований (выполняется в команде) для некого нового клиент-серверного приложения небольших размеров.
- Реализация приведенного выше, применяя технологию повторного использования для повышения степени повторного использования.

Дополнительные рекомендации по обучению:

Данный курс предоставляет хорошую возможность начать вовлекать студентов в работу с системами умеренных размеров, что позволяет студентам развивать важные навыки чтения и понимания кода, написанного другими.

Предполагается, что студенты, посещающие этот курс, имеют очень поверхностное представление о концепциях программной инженерии, однако перед этим прослушают два курса, которые дадут им хорошие знания по программированию и основам информатики. Противоположное допущение делается для SE200. В курсе SE201 рекомендуется преподавать базовое подмножество языка UML, не делая попытки охватить все его возможности.

Вместо OCL преподаватель может выбрать введение в другие формальные методы моделирования.

Общее покрытие SEEK: 34 часа

CMP.ct (4 осн. часа из 20) – Технологии разработки
  CMP.ct.1 – Проектирование и использование API
  CMP.ct.2 – Библиотеки и повторное использование кода
  CMP.ct.3 – Объектно-ориентированные аспекты времени выполнения
FND.ef (3 осн. часа из 23) – Инженерные основы программного обеспечения
  FND.ef.1 – Эмпирические и экспериментальные методы
  FND.ef.4 – Разработка систем
  FND.ef.5 – Инженерное проектирование
PRF.pr (1 осн. час из 20) – Профессионализм
MAA.md (2 осн. часа из 19) – Основы моделирования
  MAA.md.1 – Принципы моделирования
  MAA.md.2 – Пред- и постусловия, инварианты

MAA.md.3 – Введение в языки математического моделирования и языки описания спецификаций
MAA.tm (1 осн. час из 12) – Типы моделей
MAA.rfd (1 осн. час из 3) – Основы управления требованиями
MAA.er (1 осн. час из 4) – Выявление требований
MAA.rsd (1 осн. час из 6) – Документирование и спецификация требований
MAA.rsd.3 – Языки описания спецификаций
MAA.rv (1 осн. час из 3) – Проверка требований
DES.con (2 осн. часа из 3) – Концепции проектирования программного обеспечения
DES.str (3 осн. часа из 6) – Стратегии проектирования программного обеспечения
DES.ar (2 осн. часа из 9) – Архитектурное проектирование
DES.hci (1 осн. час из 12) – Проектирование человеко-машинного интерфейса
DES.dd (2 осн. часа из 12) – Детальное проектирование
DES.nst (1 осн. час из 3) – Нотации и средства поддержки проектирования
DES.ev (1 осн. час из 3) – Оценка дизайна
VAV.fnd (1 осн. час из 5) – Терминология и основы верификации и аттестации программного обеспечения
VAV.rev (1 осн. час из 6) – Рецензии кода
VAV.tst (2 осн. часа из 21) – Тестирование
VAV.par (1 осн. час из 4) – Анализ проблем и создание отчетов
PRO.imp (1 осн. час из 10) – Внедрение процессов
MGT.con (1 осн. час из 2) – Концепции менеджмента

SE211. Конструирование программного обеспечения (Software construction)

Данный курс является частью вводного блока по программной инженерии I (Core Software Engineering Package I) и предназначен для позиции A в шаблоне учебного плана.

Описание курса:

Общие принципы дисциплинированного детального проектирования программного обеспечения. БНФ (нормальная форма Бэкуса-Наура) и основы теории грамматик и синтаксического анализа. Использование генераторов синтаксических анализаторов. Основы проектирования языков и протоколов. Формальные языки. Проектирование программного обеспечения на основе таблиц/состояний. Формальные методы разработки программного обеспечения. Методы управления параллелизмом и коммуникацией между процессами. Методы проектирования программного обеспечения для численных расчетов. Средства проектирования, основанные на моделях. Введение в промежуточное программное обеспечение (middleware). Поиск критичных по времени участков и оптимизация производительности.

Требования к слушателям: SE201 или SE200, CS103 and CS105.

Задачи обучения:

Студенты, успешно прослушавшие данный курс, должны уметь:

- использовать широкий выбор методов и средств разработки программного обеспечения, включая методы, основанные на состояниях, и основанные на табличном описании подходы к детальному проектированию программного обеспечения;
- проектировать простые языки и протоколы, подходящие для различных приложений;
- генерировать код для простых языков и протоколов, используя подходящие средства;
- создавать простые формальные спецификации низкоуровневых модулей, проверять правильность этих спецификаций и генерировать для них код, используя подходящие средства;
- проектировать простое параллельное программное обеспечение;
- анализировать программное обеспечение для улучшения его производительности, надежности и сопровождаемости.

Предлагаемая последовательность преподаваемых модулей:

1. Основы формальных языков. Синтаксис и семантика. Грамматики. Формы Бэкуса-Наура. Синтаксический анализ.
2. Лексический анализ, лексемы, регулярные выражения и сети переходов. Принципы работы сканеров (лексических анализаторов).
3. Использование средств генерации сканеров, применение сканеров. Связь сканеров с компиляторами.
4. Концепции синтаксического анализа. Деревья синтаксического анализа. Контекстно-свободные грамматики, LL-разбор.
5. Обзор принципов языков программирования. Критерии для выбора языков программирования и платформ.
6. Средства автоматического проектирования и разработки программного обеспечения. Моделирование поведения системы с помощью расширенных конечных автоматов.
7. SDL.
8. Использование параллелизма и анализ параллельных архитектур.

Примеры лабораторных заданий:

- Использование средств программной инженерии для проектирования.
- Использование генераторов синтаксических анализаторов для генерации языков.

Дополнительные рекомендации по обучению:

Студенты должны прослушивать этот курс, имея базовые знания по конечным автоматам и параллелизму; данный курс покрывает углубленный материал.

Общее покрытие SEEK: 36 часов

CMP.ct (10 осн. часов из 20) – Технологии разработки

CMP.ct.6 – Обработка ошибок, обработка исключений, отказоустойчивость
CMP.ct.7 –Методы построения, основанные на состояниях и управляемые таблицами
CMP.ct.8 – Конфигурация времени выполнения и интернационализация
CMP.ct.9 – Обработка данных на основе грамматик
CMP.ct.10 – Базовые конструкции параллелизма
CMP.ct.11 – Промежуточное программное обеспечение
CMP.ct.12 – Методы построения распределенных программных систем
CMP.ct.14 – Поиск критичных по времени участков и оптимизация производительности
CMP.tl (3 осн. часа из 4) – Средства разработки
CMP.fm (8 осн. часов из 8) – Формальные методы разработки программного обеспечения
FND.mf (11 осн. часов из 56) – Основы математики
FND.mf.5 (1 осн. час из 5) – Графы и деревья
FND.mf.7 (4 осн. часа из 4) – Конечные автоматы, регулярные выражения
FND.mf.8 (4 осн. часа из 4) – Грамматики
FND.mf.9 (2 осн. часа из 4) – Вычислительная точность, погрешность и ошибки
MAA.md (4 осн. часа из 19) – Основы моделирования

SE212. Подход программной инженерии к человеко-машинному взаимодействию

Данный курс является частью вводных блоков по программной инженерии I и II (Core Software Engineering Package I and II). Он предназначен для позиции B в шаблоне учебного плана.

Описание курса:

Психологические принципы человеко-машинного взаимодействия. Оценка интерфейсов пользователя. Проектирование удобства использования. Анализ задач, проектирование и прототипирование, ориентированные на пользователя. Концептуальные модели и метафоры. Обоснование проектных решений. Проектирование окон, меню и команд. Ввод/вывод с использованием голоса и естественных языков. Время отклика и обратная связь. Цвета, пиктограммы и звук. Интернационализация и локализация. Архитектуры пользовательского интерфейса и API. Учебные примеры и проект.

Требования к слушателям: SE201 или SE200.

Задачи обучения:

Студенты, успешно прослушавшие данный курс, должны уметь:

- оценивать пользовательские интерфейсы, используя эвристическое оценивание и методы наблюдения за пользователем;

- проводить простые формальные эксперименты по оценке эргономических гипотез;
- применять ориентированное на пользователя проектирование и принципы эргономики при проектировании широкого круга программных пользовательских интерфейсов.

Предлагаемая последовательность преподаваемых модулей:

1. Основы человеко-машинного взаимодействия. Обоснование с точки зрения психологии и когнитивных наук.
2. Подробнее об основах. Методы оценивания: эвристическая оценка.
3. Подробнее о методах оценивания: записанное на видео тестирование пользователем, когнитивные просмотры (cognitive walkthroughs).
4. Анализ задач. Проектирование, ориентированное на пользователя.
5. Процесс разработки удобных в использовании приложений. Проведение экспериментов.
6. Концептуальные модели и метафоры.
7. Проектирование пользовательских интерфейсов: методы программирования цветовых схем, шрифтов, звука, анимации и т.д.
8. Проектирование пользовательских интерфейсов: размещение объектов на экране, время отклика, обратная связь, шрифты, сообщения об ошибках и т.д.
9. Проектирование пользовательских интерфейсов для специальных устройств. Использование голосового ввода/вывода.
10. Проектирование пользовательских интерфейсов: интернационализация, система помощи и т.д. Программные архитектуры пользовательских интерфейсов.
11. Обоснование проекта при проектировании пользовательского интерфейса.

Примеры лабораторных заданий:

- Эвристическое оценивание пользовательского интерфейса.
- Оценка пользовательского интерфейса с использованием видеозаписи действий пользователя.
- Прототипирование пользовательского интерфейса на бумаге с последующим обсуждением вариантов и формированием согласованного проекта.
- Писательский семинар (writers workshop) по критическому обсуждению стиля прототипов, представленных другими.
- Проектирование пользовательского интерфейса методом быстрого прототипирования.

Дополнительные рекомендации по обучению:

- Некоторым студентам трудно выявить потребности пользователей, в то время как другие находят материал курса настолько интуитивно понятным, что чувствуют себя слишком самоуверенно. Студентов следует научить учитывать мнение пользователей при оценке пользовательского интерфейса.

- Стратегия, которая хорошо работает для данного курса, состоит в обучении технологическим вопросам во время одной лекции каждую неделю и вопросам проектирования во время другой лекции каждую неделю. В сущности, читаются два параллельных курса.
- При обсуждении анализа задач его следует сравнить с анализом вариантов использования.
- Формат «писательских семинаров» (writers workshops) удачно подходит для обучения проектированию. В таких семинарах обычно участвуют три группы студентов. Первая представляет прототипы пользовательских интерфейсов проектируемого приложения на бумаге. Затем вторая группа рассказывает, что им нравится в этом прототипе. И, наконец, третья группа предлагает конструктивную критику.

Общее покрытие SEEK: 25 часов

CMP.ct (1 осн. час из 20) – Технологии разработки
CMP.ct.8 – Конфигурация времени выполнения и интернационализация
CMP.tl.2 – Среды разработки графического интерфейса пользователя (GUI)
FND.ef (3 осн. часа из 23) – Инженерные основы программного обеспечения
PRF.psy (1 осн. час из 5) – Групповая динамика и психология
MAA.md (4 осн. часа из 19) – Основы моделирования
MAA.tm (1 осн. час из 12) – Типы моделей
MAA.rfd.5 – Анализ качества
DES.hci (6 осн. часов из 12) – Проектирование человеко-машинного интерфейса
VAV.fnd (1 осн. час из 5) – Терминология и основы верификации и аттестации программного обеспечения
VAV.fnd.4 -Метрики и измерения
VAV.rev (1 осн. час из 6) – Рецензии кода
VAV.rev.3 – Инспекции
VAV.tst.9 – Тестирование атрибутов качества
VAV.hct (6 осн. часов из 6) – Тестирование и оценка человеко-машинного интерфейса
QUA.pda (1 осн. час из 4) – Обеспечение качества продукта
QUA.pda.6 – Оценка атрибутов качества продукта

SE213. Проектирование и архитектура больших систем программного обеспечения

Данный курс является частью вводного блока по программной инженерии II (Core Software Engineering Package II) и предназначен для позиции А в шаблоне учебного плана.

Описание курса:

Моделирование и проектирование гибкого программного обеспечения на архитектурном уровне. Основы модельно-управляемой архитектуры. Архитектурные стили и шаблоны. Промежуточное программное обеспечение (middleware) и

среды разработки приложений (application frameworks). Виды конфигураций и управление конфигурациями. Линейки программных продуктов (product lines). Проектирование с использованием «коробочного» программного обеспечения (Commercial Off-The-Shelf software, COTS).

Требования к слушателям: (SE201 или SE200) и CS103.

Задачи обучения:

Студенты, успешно прослушавшие данный курс, должны уметь:

- используя требования, описывающие простые системы, разрабатывать программные архитектуры и высокоуровневые проекты;
- эффективно использовать средства управления конфигурациями и правильно применять процессы управления изменениями;
- проектировать простые распределенные системы программного обеспечения;
- проектировать программное обеспечение с использованием «коробочного» программного обеспечения.
- применять различные среды и архитектуры при проектировании разнообразных типов программного обеспечения.
- проектировать и реализовывать программное обеспечение, используя несколько различных технологий промежуточного программного обеспечения.

Дополнительные аспекты обучения:

Данный курс предшествует изучению детального проектирования. Поэтому студентам необходимы средства и пакеты программ, которые позволяют реализовать их проект, не заботясь о низкоуровневых деталях.

Общее покрытие SEEK: 28 часов

MAA.md (5 осн. часов из 19) – Основы моделирования

MAA.tm (5 осн. часов из12) – Типы моделей

DES.str (2 осн. часа из 6) – Стратегии проектирования программного обеспечения

DES.ar (5 осн. часов из 9) – Архитектурное проектирование

EVO.pro (3 осн. часа из 6) – Процессы эволюции

EVO.pro.1 – Основные концепции эволюции и сопровождения

EVO.pro.2 – Взаимодействие между эволюционирующими объектами

EVO.ac (2 осн. часа из 4) – Деятельность по обеспечению эволюции

MGT.con (1 осн. час из 2) – Концепции менеджмента

MGT.pp (1 осн. час из 6) – Планирование проекта

MGT.cm (4 осн. часа из 5) – Управление конфигурациями программного обеспечения

SE221. Тестирование программного обеспечения

Данный курс является частью вводного блока по программной инженерии II (Core Software Engineering Package II) и предназначен для позиции C в шаблоне учебного плана.

Описание курса:

Методы и принципы тестирования: дефекты (defect) и отказы (failure), классы эквивалентности, граничное тестирование. Типы дефектов. Тестирование методом черного ящика и структурное тестирование. Стратегии тестирования: модульное тестирование (unit testing), интеграционное тестирование, профилирование, разработка, управляемая тестированием (test-driven development). Тестирование, основанное на состояниях. Тестирование конфигурации. Тестирование совместимости. Тестирование web-сайтов. Альфа-, бета- и приемочное тестирование. Критерии покрытия. Инструментарий и средства тестирования. Разработка планов тестирования. Управление процессами тестирования. Отслеживание проблем, анализ и отчетность.

Требования к слушателям: SE201 или SE200.

Задачи обучения:

Студенты, успешно прослушавшие данный курс, должны уметь:

- анализировать требования для определения подходящей стратегии тестирования;
- проектировать и реализовать планы комплексного тестирования;
- эффективно и квалифицированно применять разнообразные методы тестирования;
- рассчитывать тестовое покрытие и результативность тестирования согласно множеству критериев;
- использовать статистические методы для оценивания плотности дефектов и вероятности отказов;
- проводить рецензии кода и инспекции.

Дополнительные аспекты обучения:

Данный курс покрывает 95% знаний по тестированию, включая разнообразные методы тестирования.

Данный курс должен развить практические навыки и дать опыт студенту, желательно путем участия в разработке реальных программ.

Тестирование эргономических характеристик и удобства использования описано в SE212.

Общее покрытие SEEK: 23 часа

MAA.rfd (1 осн. час из 3) – Основы управления требованиями
MAA.rfd.4 – Характеристики требований

VAV.fnd (2 осн. часа из 5) – Терминология и основы верификации и аттестации

VAV.rev (1 осн. час из 6) – Рецензии кода

VAV.tst (14 осн. часов из 21) – Тестирование

VAV.tst.2 – Обработка исключительных ситуаций
VAV.par (3 осн. часа из 4) – Анализ проблем и создание отчетов
QUA.pda (2 осн. часа из 4) – Обеспечение качества продукта

SE311. Проектирование и архитектура программного обеспечения

Данный курс является частью вводного блока по программной инженерии II (Core Software Engineering Package II) и предназначен для позиции D в шаблоне учебного плана.

Описание курса:

Углубленное изучение проектирования программного обеспечения. Продолжение изучения шаблонов проектирования, сред разработки и архитектур. Исследование существующих архитектур промежуточного программного обеспечения. Проектирование распределенных систем с использованием промежуточного программного обеспечения. Компонентное проектирование. Теория измерения и использование метрик в проектировании. Проектирование с учетом таких качеств, как производительность, безопасность, защищенность, возможность повторного использования, надежность и т.д. Измерение внутренних параметров и сложности программного обеспечения. Оценка и эволюция проектирования. Основы эволюции программного обеспечения, реинжиниринга и обратной инженерии.

Требования к слушателям: SE211.

Задачи обучения:

Студенты, успешно прослушавшие данный курс, должны уметь:

- применять множество различных шаблонов проектирования, сред разработки и архитектур в проектировании разнообразного программного обеспечения;
- проектировать и реализовывать программное обеспечение, используя несколько различных технологий промежуточного программного обеспечения;
- использовать адекватные метрики качества как средство оценки качества проектирования, оценивать соответствие результатов проектирования поставленным целям;
- модифицировать проекты, используя продуманные подходы к управлению изменениями;
- использовать методы обратной инженерии (reverse engineering) для восстановления дизайна программного обеспечения .

Рекомендуемая последовательность преподаваемых модулей:

1. Углубленное изучение шаблонов проектирования (design patterns), построенное на ранее изученном материале
2. Примеры применения шаблонов проектирования к прикладным задачам
3. Углубленное изучение архитектур промежуточного программного обеспечения, включая COM, CORBA и .NET
4. Расширенные примеры реальных проектов

5. Основы метрик программного обеспечения; измерение показателей качества программного обеспечения
6. Методы реинжиниринга и обратной инженерии

Примеры лабораторных заданий:

- Построение значительного проекта с использованием одной или более хорошо известных архитектур промежуточного программного обеспечения.

Дополнительные аспекты обучения:

Подразумевается, что студенты знакомы с некоторыми основными шаблонами проектирования; данный курс подробно охватывает известный на сегодня спектр шаблонов, не ограничиваясь классическими шаблонами «Gang of Four».

Общее покрытие SEEK: 33 часа

CMP.ct (3 осн. часа из 20) – Технологии разработки
CMP.ct.11 – Промежуточное программное обеспечение
CMP.ct.12 – Методы построения распределенной программной системы
CMP.ct.13 – Построение гетерогенных систем
MAA.md (4 осн. часа из 19) – Основы моделирования
MAA.tm.3 – Структурное моделирование
DES.str (2 осн. часа из 6) – Стратегии проектирования программного обеспечения
DES.ar (5 осн. часов из 9) – Архитектурное проектирование
DES.dd (8 осн. часов из 12) – Детальное проектирование
DES.nst (1 осн. час из 3) – Нотации и средства поддержки проектирования
DES.ev (1 осн. час из 3) – Оценка дизайна
EVO.pro (5 осн. часов из 6) – Процессы эволюции
EVO.ac (4 осн. часа из 4) – Деятельность по обеспечению эволюции

SE312. Детальное проектирование программного обеспечения

Данный курс является частью вводного блока по программной инженерии II (Core Software Engineering Package II) и предназначен для позиции D в шаблоне учебного плана.

Описание курса:

Углубленное изучение детального проектирования и разработки программного обеспечения. Подробное изучение шаблонов проектирования и рефакторинга. Введение в формальные подходы к проектированию. Анализ проектов, основанный на внутренних критериях качества. Улучшение производительности и простоты поддержки. Обратная инженерия. Дисциплина изменения архитектуры проекта.

Требования к слушателям: SE213.

Задачи обучения:

Студенты, успешно прослушавшие данный курс, должны уметь:

- применять множество различных средств и методов разработки программного обеспечения, включая основанные на состояниях и табличные подходы к детальному проектированию программного обеспечения;
- использовать множество различных шаблонов проектирования в проектировании программного обеспечения;
- выполнять объектно-ориентированное проектирование и программирование на высоком профессиональном уровне;
- анализировать программное обеспечение для повышения его эффективности, надежности и простоты сопровождения;
- модифицировать проекты, используя продуманные подходы к управлению изменениями;
- использовать методы возвратного проектирования для воссоздания проекта программного обеспечения.

Дополнительные аспекты обучения:

К данному моменту студенты должны знать достаточно большой объем материала по высокоуровневому проектированию и архитектуре. Данный курс покрывает низкоуровневые детали.

Общее покрытие SEEK: 26 часов

CMP.ct (13 осн. часов из 20) – Технологии разработки

CMP.tl (3 осн. часа из 4) – Средства разработки

CMP.fm (2 осн. часа из 8) – Формальные методы разработки программного обеспечения MAA.tm (2 осн. часа из 12) – Типы моделей

DES.dd (5 осн. часов из 12) – Детальное проектирование

EVO.ac (1 осн. час из 4) – Деятельность по обеспечению эволюции

SE313. Формальные методы программной инженерии

Данный курс является частью вводного блока по программной инженерии II (Core Software Engineering Package II) и предназначен для позиции F в шаблоне учебного плана.

Описание курса:

Обзор математических основ формальных методов. Формальные языки и методы описания спецификаций и проектов, включая описания синтаксиса с использованием грамматик и конечных автоматов. Анализ и верификация спецификаций и проектов. Использование утверждений (assertion) и доказательств. Автоматизированные преобразования программы и дизайна.

Требования к слушателям: SE312.

Задачи обучения:

Студенты, успешно прослушавшие данный курс, должны уметь:

- создавать математически точные спецификации и проекты, используя такие языки, как OCL, Z и т.д.;
- анализировать свойства формальных спецификаций и проектов;
- использовать инструменты преобразования спецификаций и проектов.

Общее покрытие SEEK: 34 часа

CMP.fm (6 осн. часов из 8) – Формальные методы разработки программного обеспечения

FND.mf (13 осн. часов из 56) – Основы математики

FND.mf.5 (1 осн. час из 5) – Графы и деревья

FND.mf.7 (4 осн. часа из 4) – Конечные автоматы, регулярные выражения

FND.mf.8 (4 осн. часа из 4) – Грамматики

FND.mf.9 (4 осн. часа из 4) – Вычислительная точность, погрешность и ошибки

MAA.md (3 осн. часа из 19) – Основы моделирования

MAA.md.3 – Введение в математические модели и языки описания спецификаций

MAA.tm (2 осн. часа из 12) – Типы моделей

MAA.tm.2 – Поведенческое моделирование

MAA.rsd (3 осн. часа из 6) – Спецификация и документирование требований

MAA.rsd.3 – Языки описания спецификаций

MAA.rv (1 осн. час из 3) – Проверка требований

DES.dd (3 осн. часа из 12) – Детальное проектирование

DES.nst (1 осн. час из 3) – Нотации и средства проектирования

DES.nst.6 – Формальный анализ проекта

DES.ev (1 осн. час из 3) – Оценка дизайна

DES.ev.2 – Методы оценки

EVO.ac (1 осн. час из 4) – Деятельность по обеспечению эволюции

EVO.ac.6 – Рефакторинг

EVO.ac.7 – Преобразование программ

SE321. Обеспечение качества и тестирование программного обеспечения

Данный курс является частью вводного блока по программной инженерии I (Core Software Engineering Package I) и предназначен для позиции C в шаблоне учебного плана.

Описание курса:

Качество: как его обеспечить и верифицировать; необходимость культуры качества. Предотвращение ошибок и проблем. Проверки инспекциями и рецензии. Методы тестирования, верификации и аттестации. Качество процесса в сопоставлении с качеством продукта. Стандарты качества. Обеспечение качества продукта и качества процесса. Анализ и отчетность по проблемам. Статистические подходы к контролю качества.

Требования к слушателям: SE201 или SE200.

Задачи обучения:

Студенты, успешно прослушавшие данный курс, должны уметь:

- осуществлять эффективные и квалифицированные инспекции;
- проектировать и реализовывать планы по комплексному тестированию;

- эффективно и квалифицированно применять всевозможные методы тестирования
- рассчитывать покрытие и результативность тестирования на основании множества критериев;
- использовать статистические методы для оценивания плотности дефектов и вероятности отказов;
- инспектировать процесс разработки программного обеспечения с целью оценки эффективности контроля качества.

Рекомендуемая последовательность преподаваемых модулей:

1. Введение в обеспечение качества программного обеспечения.
2. Инспекции и рецензии.
3. Принципы аттестации программного обеспечения.
4. Верификация программного обеспечения.
5. Тестирование программного обеспечения.
6. Методы построения тестов, основанные на спецификациях.
7. Тестирование методами «белого ящика» и «серого ящика».
8. Методы построения тестов, основанные на управляющей логике.
9. Методы построения тестов, основанные на потоках данных.
10. Подход «чистой комнаты» (cleanroom approach) к обеспечению качества.
11. Сертификация процесса разработки программного обеспечения.

Примеры лабораторных заданий:

- Использование автоматизированных средств тестирования.
- Тестирование различных видов программного обеспечения.
- Применение различных методов тестирования.
- Коллективное инспектирование программного обеспечения; сравнение и анализ результатов.

Дополнительные аспекты обучения:

Тестирование пользовательского интерфейса описано в курсе SE212, в данный курс эта тема не включена. Несмотря на это, использование тестовых пакетов (test harnesses), проверяющих работу пользовательского интерфейса, является подходящей темой для включения в курс.

Причина такого внимания к тестированию не в том, что остальные методы менее важны, а в том, что многие другие методы контроля качества (например, инспекции) проще изучить в процессе работы, в то время как материал по тестированию лучше включить в отдельный курс.

Общее покрытие SEEK: 37 часов

FND.mf (2 осн. часа из 56) – Основы математики

FND.mf.9 (2 осн. часа из 4) – Вычислительная точность, погрешность и ошибки

VAV.fnd (2 осн. часа из 5) – Терминология и основы верификации и аттестации программного обеспечения

VAV.rev (1 осн. час из 6) – Рецензии кода

VAV.tst (14 осн. часов из 21) – Тестирование

VAV.par (3 осн. часа из 4) – Анализ проблем и создание отчетов

PRO.con (1 осн. час из 3) – Основные концепции процессов разработки ПО

QUA.cc (1 осн. час из 2) – Концепции и культура качества программного обеспечения

QUA.std (2 осн. часа из 2) – Стандарты качества программного обеспечения

QUA.pro (4 осн. часа из 4) – Процессы управления качеством программного обеспечения

QUA.pca (4 осн. часа из 4) – Обеспечение качества процесса

QUA.pda (3 осн. часа из 4) – Обеспечение качества продукта

SE322. Анализ требований к программному обеспечению

Данный курс является частью вводного блока по программной инженерии I (Core Software Engineering Package I) и предназначен для позиции E в шаблоне учебного плана.

Описание курса:

Инженерия предметной области. Методы выявления и формализации требований. Языки и модели представления требований. Методы анализа и аттестации, включая анализ потребностей, целей и примеров использования. Системные требования. Определение и измерение внешних качеств, таких как производительность, надежность, доступность, безопасность, защищенность и т.д. Определение и анализ требований к различным типам систем: встроенных систем, потребительских систем, web-систем, бизнес-систем, научных систем и других инженерных систем. Разрешение конфликтов функциональности системы. Стандарты документирования требований. Отслеживание требований.

Человеческие факторы. Требования в контексте гибких процессов. Управление требованиями; управление изменениями требований.

Требования к слушателям: SE201 или SE200.

Задачи обучения:

Студенты, успешно прослушавшие данный курс, должны уметь:

- обнаруживать или выявлять требования, используя различные методы;
- приоритезировать требования;
- применять методы анализа, такие как анализ потребностей, анализ целей и варантов использования;
- оценивать требования в соответствии с критериями выполнимости, ясности, отсутствием неоднозначностей и т.д.;
- представлять функциональные и нефункциональные требования для различных типов систем, используя формальные и неформальные методы;
- определять и измерять атрибуты качества;
- вести переговоры с различными заинтересованными лицами для достижения согласия по множеству требований;

- обнаруживать и разрешать проблемы взаимодействия элементов.

Рекомендуемая последовательность преподаваемых модулей:

1. Основы инженерии требований к программному обеспечению.
2. Процессы инженерии требований: выявление требований, спецификация, анализ и управление.
3. Типы требований: функциональные, нефункциональные, атрибуты качества.
4. Выявление требований: определение потребностей, целей и требований. Заказчики и другие заинтересованные лица. Опросы и наблюдения.
5. Спецификация требований: текстовые и графические нотации и языки (UML, нотации пользовательских требований). Методы написания высококачественных требований. Стандарты документирования.
6. Анализ требований: инспекция, аттестация, завершенность, обнаружение конфликтов и несоответствий. Анализ взаимодействия элементов функциональности (feature interaction) и разрешение противоречий.
7. Целенаправленное и ориентированное на варианты использования моделирование, прототипирование и методы анализа.
8. Требования к типичным системам: встроенным системам, потребительским системам, web-системам, бизнес-системам, научным системам и другим инженерным системам.
9. Управление требованиями: отслеживание, приоритеты, изменения, базовые линии и инструментальная поддержка.
10. Согласование требований и управление рисками.
11. Интеграция анализа требований и процессов разработки программного обеспечения (включая Agile-процессы).

Примеры лабораторных заданий:

- Четкое описание требований.
- Анализ множества всевозможных существующих программных систем: измерение качества и восстановление требований (reverse engineering) по программе.
- Опрос пользователей и итеративное транслирование результатов опроса в прототипы.
- Использование инструментов управления требованиями.
- Моделирование, прототипирование и анализ требований с помощью средств UML/URN.
- Разрешение конфликтов при взаимодействии различных функций.

Дополнительные аспекты обучения:

Преподаватели данного курса должны будут приложить определенные усилия для мотивации к обучению студентов, увлеченных скорее технической стороной программной инженерии и программированием. Будет полезно указать примеры плохих требований, приведших к потерям (экономическим или физиче-

ским). Также будет полезно взаимодействие с реальным или искусственно сымитированным заказчиком.

Общее покрытие SEEK: 18 часов

MAA.tm (9 осн. часов из 12) – Типы моделей

MAA.rfd (1 осн. час из 3) – Основы управления требованиями

MAA.er (2 осн. часа из 4) – Выявление требований

MAA.rsd (4 осн. часа из 6) – Спецификация и документация требований

MAA.rv (1 осн. час из 3) – Проверка требований

MAA.rfd.6 (1 осн. час из 3) – Управление требованиями

SE323. Управление программными проектами

Данный курс является частью вводного блока по программной инженерии I (Core Software Engineering Package I) и предназначен для позиции F в шаблоне учебного плана.

Описание курса:

Планирование проекта, оценка стоимости и составление расписания. Средства управления проектом. Факторы, влияющие на продуктивность и успех. Метрики продуктивности. Анализ вариантов и рисков. Планирование изменений. Управление ожиданиями. Управление выпуском продукции и конфигурацией. Стандарты процесса разработки и реализации процесса. Договорные соглашения и интеллектуальная собственность. Подходы к сопровождению и долгосрочной разработке программного обеспечения. Примеры реальных производственных проектов.

Требования к слушателям: SE321 и SE322.

Задачи обучения:

Студенты, успешно прошедшие данный курс, должны уметь:

- разрабатывать всеобъемлющий план проекта, требующего значительного объема работ;
- применять методы управления к проектам, следующим как «гибким», так и «традиционным» методологиям;
- эффективно оценивать проектные затраты, используя несколько различных методов;
- применять методы измерений, основанные на количестве точек входа (function points);
- измерять прогресс проекта, продуктивность и другие аспекты процесса разработки программного обеспечения;
- применять методы анализа заработанной стоимости (earned-value analysis);
- управлять рисками, динамично регулируя планы проекта;
- эффективно использовать средства управления конфигурациями и правильно применять процессы управления изменениями;

- разрабатывать и оценивать простые лицензионные соглашения на использование программного обеспечения, контракты и договоры об интеллектуальной собственности, в то же время осознавая необходимость привлечения юриста для такого рода работы;
- использовать стандарты в управлении проектами, включая ISO 10006 (качество управления проектом) и ISO 12207 (процесс программного обеспечения) вместе с моделью SEI CMM

Рекомендуемая последовательность преподаваемых модулей:

1. Основные концепции управления проектами.
2. Управление требованиями.
3. Жизненный цикл программного обеспечения.
4. Оценивание программного обеспечения .
5. План проекта.
6. Мониторинг проекта.
7. Анализ рисков.
8. Управление качеством.
9. Человеческие факторы.

Примеры лабораторных заданий:

- Использование коммерческого средства управления проектом для всесторонней поддержки управления программным проектом. Это включает в себя создание диаграмм Ганта, PERT и приобретенной стоимости.
- Оценка стоимости затрат для малых систем, используя различные методы.
- Разработка плана проекта для большой системы.
- Написание плана управления конфигурацией.
- Использование средств контроля за изменениями и управлением конфигурацией.
- Оценивание программного контракта или лицензии.

Общее покрытие SEEK: 26 часов

MAA.mgt (2 осн. часа из 3) – Управление требованиями
PRO.con (2 осн. часа из 3) – Основные концепции процессов разработки ПО
PRO.imp (9 осн. часов из 10) – Внедрение процессов
MGT.con (1 осн. час из 2) – Концепции менеджмента
MGT.pp (3 осн. часа из 6) – Планирование проекта
MGT.per (1 осн. час из 2) – Организация и управление персоналом
MGT.ctl (4 осн. часа из 4) – Контроль над проектом
MGT.cm (4 осн. часа из 5) – Управление конфигурациями программного обеспечения

SE324. Процесс разработки и управления программным обеспечением

Данный курс является частью вводного блока по программной инженерии II (Core Software Engineering Package II) и предназначен для позиции E в шаблоне учебного плана.

Описание курса:

Процессы разработки программного обеспечения: стандарты, реализация и обеспечение. Управление проектами, сфокусированное на управлении требованиями и долговременной эволюции: выявление требований и определение их приоритетов, оценка стоимости, планирование и отслеживание проекта, анализ рисков, контроль над проектом, управление изменениями.

Требования к слушателям: SE201 или SE200, плюс как минимум два дополнительных курса по программной инженерии второго уровня или выше.

Задачи обучения:

Студенты, успешно прослушавшие данный курс, должны уметь:

- выявлять требования, используя различные методы;
- формировать требования и определять их приоритеты;
- проектировать процессы, подходящие к различным типам проектов;
- оценивать процесс разработки программного обеспечения с целью определения его влияния на качество;
- разрабатывать всеобъемлющий план проекта для большого объема работ;
- измерять прогресс проекта, продуктивность и другие аспекты процесса разработки программного обеспечения;
- эффективно рассчитывать расходы на развертывание и эволюцию системы, используя различные методы;
- выполнять управление рисками, динамически изменяя план проекта;
- использовать стандарты управления качеством, процессом разработки и проектом;
- выполнять анализ первопричин проблем и работать в направлении непрерывного улучшения процесса.

Общее покрытие SEEK: 39 часов

MAA.er (2 осн. часа из 4) – Выявление требований
MAA.rsd (1 осн. час из 6) – Документирование и спецификация требований
MAA.rfd.6 (3 осн. часа из 3) -Управление требованиями
EVO.pro (2 осн. часа из 6) – Процессы эволюции
EVO.pro.3 – Модели эволюции программного обеспечения
EVO.pro.4 – Модели затрат на эволюцию программного обеспечения
PRO.con (3 осн. часа из 3) – Основные концепции процессов разработки ПО
PRO.imp (9 осн. часов из 10) – Внедрение процессов
QUA.cc (1 осн. час из 2) – Концепции и культура качества программного обеспечения
QUA.std (2 осн. часа из 2) – Стандарты качества программного обеспечения

QUA.pro (4 осн. часа из 4) – Процессы управления качеством программного обеспечения
QUA.pca (4 осн. часа из 4) – Обеспечение качества процесса
QUA.pda (1 осн. час из 4) – Обеспечение качества продукта
MGT.pp (2 осн. часа из 6) -Планирование проекта
MGT.per (1 осн. час из 2) – Организация и управление персоналом
MGT.ctl (4 осн. часа из 4) – Контроль над проектом

Дипломный проект (Capstone Project)

SE400. Дипломный проект по программной инженерии

Дипломный (завершающий) проект являлся частью учебного плана по инженерии с тех давних времен, когда каменщику предлагали вырезать красивый «замковый» камень (обычно использовался как завершающий элемент карниза или свода печи), для демонстрации уровня своего мастерства.

Описание курса:

Разработка крупной программной системы, применяя знания, полученные из курсов программы. Содержит разработку требований, проектирование, реализацию и обеспечение качества. Студенты могут придерживаться любой подходящей модели процесса, должны уделять внимание вопросам качества и должны управлять проектом сами, следуя всем соответствующим методам управления проектами. Успех проекта определяется большей частью тем, адекватно ли разрешили студенты задачу, заданную им заказчиком.

Требования к слушателям: окончание курсов третьего уровня по одному из шаблонов учебного плана.

Примерный комплект результатов:

Ожидается, что студенты проведут одну или несколько итераций разработки программной системы, вместе со всеми артефактами, соответствующими модели процесса, которую они используют. Сюда желательно включить план проекта (возможно регулярно обновляющийся и содержащий оценку стоимости, анализ рисков, разбиение работы по заданиям и т.д.), требования (включая варианты использования), архитектурную и проектную документации, планы тестирования, исходный код и устанавливаемую систему.

Дополнительные аспекты обучения:

- Допускается, что курс не будет излагаться в виде формальных лекций, несмотря на то, что студенты, вероятно, будут посещать рабочие презентации других групп.
- Рекомендуется, чтобы у студентов был свой «заказчик», для которого они будут разрабатывать программное обеспечение. Это может быть компания, или профессор, или несколько человек, которые достаточно адекватно представляют потенциальный рынок. Целью проекта может быть

решение задачи заказчика, поэтому заказчик мог бы ассистировать педагогу при оценке работы.

- Настоятельно рекомендуется работа над дипломными проектами студентов в группах как минимум из двух человек, а желательно – из трех-четырех. При этом нужно разрабатывать стратегии для ситуаций, когда индивидуальный вклад в групповую работу является неодинаковым.
- Некоторые учебные заведения могут пожелать разделить данный курс на две части, например по одной в семестр. В таком случае, если студенты не заканчивают проект (т.е. второй из двух курсов), рекомендуется, чтобы они начали повторную работу с первого из двух курса.

Общее покрытие SEEK: 28 часов

Данный материал представляет модули SEEK, которые должны практиковаться во всех дипломных проектах. Помимо этого, различные проекты будут развивать навыки в различных областях SEEK.

CMP.ct (1 осн. час из 20) – Технологии разработки
PRF.psy (1 осн. час из 5) – Групповая динамика и психология
PRF.com (2 осн. часа из 10) – Навыки коммуникации
PRF.pr (2 осн. часа из 20) – Профессионализм
MAA.tm (1 осн. час из 12) – Типы моделей
MAA.er (1 осн. час из 4) – Выявление требований
MAA.rsd (1 осн. час из 6) – Документирование и спецификация требований
MAA.rv (1 осн. час из 3) – Проверка требований
DES.str (1 осн. час из 6) – Стратегии проектирования программного обеспечения
DES.ar (2 осн. часа из 9) – Архитектурное проектирование
DES.hci (2 осн. часа из 12) – Проектирование человеко-машинного интерфейса
DES.dd (2 осн. часа из 12) – Детальное проектирование
DES.nst (1 осн. час из 3) – Нотации и средства поддержки проектирования
DES.ev (1 осн. час из 3) – Оценка дизайна
VAV.rev (2 осн. часа из 6) – Рецензии кода
VAV.tst (3 осн. часа из 21) – Тестирование
MGT.pp (2 осн. часа из 6) – Планирование проекта
MGT.per (1 осн. час из 2) – Организация и управление персоналом
MGT.cm (1 осн. час из 5) – Управление конфигурациями программного обеспечения

Приложение Б. Участники и рецензенты проекта

Волонтеры по определению преподаваемого материала

Jonathan D. Addelston, UpStart Systems, U.S.
Roger Alexander, Colorado State University, U.S.
Niniek Angkasaputra, Fraunhofer Institute of Experimental Software Engineering, Germany
Mark A. Ardis, Rose-Hulman University, U.S.
Jocelyn Armarego, Murdoch University, Australia
Doug Baldwin, The State University of New York, Geneseo, U.S.
Earl Beede, Construx, U.S.
Fawsy Bendeck, University of Kaiserslautern, Germany
Mordechai Ben-Menachem, Ben-Gurion University, Israel
Robert Burnett, consultant, Brazil
Kai Chang, Auburn University, U.S.
Jason Chen, National Central University, Taiwan
Cynthia Cicalese, Marymount University, U.S.
Tony (Anthony) Cowling, University of Sheffield, U.K.
David Dampier, Mississippi State University, U.S.
Mel Damodaran, University of Houston, U.S.
Onur Demirors, Middle East Technical University, Turkey
Vladan Devedzic, University of Belgrade, Yugoslavia
Oscar Dieste, University of Alfonso X El Sabio, Spain
Dick Fairley Oregon Graduate Institute, U.S.
Mohamed E. Fayad, University of Nebraska, Lincoln, U.S.
Orit Hazzan, Israel Institute of Technology, Israel
Bill Hefley, Carnegie Mellon University, U.S.
Peter Henderson, Butler University, U.S.
Joel Henry, University of Montana, U.S.
Jens Jahnke, University of Victoria, Canada
Stanislaw Jarzabek, National University of Singapore, Singapore
Natalia Juristo, Universidad Politecnica of Madrid, Spain
Umit Karakas, consultant, Turkey
Atchutarao Killamsetty, JENS SpinNet, Japan
Haim Kilov, Financial Systems Architects, U.S.
Moshe Krieger, University of Ottawa, Canada
Hareton Leung, Hong Kong Polytechnic University, Hong Kong
Marta Lopez, Fraunhofer Institute of Experimental Software Engineering, Germany
Mike Lutz, Rochester Institute of Technology, U.S.
Paul E. MacNeil, Mercer University, U.S.
Mike McCracken, Georgia Institute of Technology, U.S.
James McDonald, Monmouth University, U.S.
Emilia Mendes, University of Auckland, New Zealand
Luisa Mich, University of Trento, Italy
Ana Moreno, Universidad Politecnica of Madrid, Spain

Traian Muntean, University of Marseilles, France
Keith Olson, Utah Valley State College, U.S.
Michael Oudshoorn, University of Adelaide, Australia
Dietmar Pfahl, Fraunhofer Institute of Experimental Software Engineering, Germany
Mario Piattini, University of Castilla-La Mancha, Spain
Francis Pinheiro, University of Brazil, Brazil
Valentina Plekhanova, University of Sunderland, U.K.
Hossein Saiedian, University of Kansas, U.S.
Stephen C. Schwarm, EMC, U.S.
Peraphon Sophatsathit, Chulalongkorn University, Thailand
Jennifer S. Stuart, Construx, U.S.
Linda T. Taylor, Taylor & Zeno Systems, U.S.
Richard Thayer, California State University, Sacramento, U.S.
Jim Tomayko, Carnegie Melon University, U.S.
Massood Towhidnejad, Embry-Riddle University, U.S.
Joseph E. Urban, Arizona State University, U.S.
Arie van Deursen, National Research Institute for Mathematics & Computer Science,
Netherlands
Sira Vegas, University of Madrid, Spain
Bimlesh Wadhwa, National University of Singapore, Singapore
Yingxu Wang, University of Calgary, Canada
Mary Jane Willshire, University of Portland, U.S.
Mansour Zand, University of Nebraska, Omaha, U.S.
Jianhan Zhu, University of Ulster, U.K.

Участники семинара «SE2004 SEEK Workshop»
Earl Beede, Construx, U.S.
Pierre Bourque, University of Quebec
David Budgen, Keele University, U.K.
Kai Chang, Auburn University, U.S.
Jorge L. Diaz-Herrera, Rochester Institute of Technology, U.S.
Frank Driscoll, Mitre Cooperation, U.S.
Steve Easterbrook, University of Toronto, Canada
Dick Fairley, Oregon Graduate Institute, U.S.
Peter Henderson, Butler University, U.S.
Thomas B. Hilburn, Embry-Riddle University, U.S.
Tom Horton, University of Virginia, U.S.
Cem Kaner, Florida Institute of Technology, U.S.
Haim Kilov, Financial Systems Architects, U.S.
Gideon Kornblum, Getronics, Netherlands
Rich LeBlanc, Georgia Institute of Technology, U.S.
Timothy C. Lethbridge, University of Ottawa, Canada
Bill Marion, Valparaiso University, U.S.
Yoshihiro Matsumoto, Musashi Institute of Technology, Japan
Mike McCracken, Georgia Institute of Technology, U.S.

Andrew McGettrick, University of Strathclyde, U.K.
Susan Mengel, Texas Tech University, U.S.
Traian Muntean, University of Marseilles, France
Keith Olson, Utah Valley State College, U.S.
Allen Parrish, University of Alabama, U.S.
Ann Sobel, Miami University, U.S.
Jenny Stuart, Construx, U.S.
Linda T. Taylor, Taylor & Zeno Systems, U.S.
Barrie Thompson, University of Sunderland, U.K.
Richard Upchurch, University of Massachussetts, U.S.
Frank H. Young, Rose-Hulman University, U.S.

Внутренние рецензенты SEEK
Barry Boehm, University of Southern California, U.S.
Kai H. Chang, Auburn University, U.S.
Jason Jen-Yen Chen, National Central University, Taiwan
Tony Cowling, University of Sheffield, U.K.
Vladan Devedzic, University of Belgrade, Yugoslavia
Laura Dillon, Michigan State University, U.S.
Dennis J. Frailey, Raytheon, U.S.
Peter Henderson, Butler University, U.S.
Watts Humphrey, Software Engineering Institute, U.S.
Haim Kilov, Financial Systems Architects, U.S.
Hareton Leung, Hong Kong Polytechnic University, Hong Kong
Yoshihiro Matsumoto, Information Processing Society, Japan
Bertrand Meyer, ETH, Zurich
Luisa Mich, University of Trento, Italy
James W. Moore, Mitre, U.S.
Hausi Muller, University of Victoria, Canada
Peter G. Neuman, SRI International, U.S.
David Notkin, University of Washington, U.S.
Dietmar Pfahl, Fraunhofer Institute of Experimental Software Engineering, Germany
Mary Shaw, Carnegie Mellon University, U.S.
Ian Sommerville, Lancaster University, U.K.
Peraphon Sophatsathit, Chulalongkorn University, Thailand
Steve Tockey, Construx Software, U.S.
Massood Towhidnejad, Embry-Riddle University, U.S.
Leonard Tripp, Boeing Sha red Services, U.S.

Внешние рецензенты SEEK
James P. Alstad, Hughes Space and Communications Company, USA
Niniek Angkasaputra, Fraunhofer Institute for Experimental SE, Germany
Hernan Astudillo, Financial Systems Architects, USA
Donald J. Bagert, Rose-Hulman Institute of Technology, USA
Mario R. Barbacci, Software Engineering Institute, USA
Ilia Bider, IbisSoft AB, Sweden

Grady Booch, Rational Corp, USA
Jurgen Borstler, Umea University, Sweden
Pierre Bourque, Ecole de Technologie Superieure, Montreal, Canada
David Budgen, Keele University, UK
Joe Clifton, University of Wisconsin - Platteville, USA
Kendra Cooper, The University of Texas at Dallas, USA
Tony Cowling, University of Sheffield, UK
Vladan Devedzic, University of Belgrade, Yogoslavia
Rick Duley, Edith Cowan University, Australia
Robert Dupuis, Universite de Quebec a Monteal, Canada
Juan Garbajosa, Universidad Politecnica de Madrid, Spain
Robert L. Glass, Indiana University, USA
Orit Hazzan, Technion -- Israel Institute of Technology, Israel
Hui Huang, National Institute of Standards and Technology, USA
IFIP Working Group 2.9
Joseph Kasser, University of South Australia
Khaled Khan, University of Western Sydney, Australia
Peter Knoke, University of Alaska, Fairbanks, USA
Gideon Kornblum, CManagement bv, Netherlands
Claude Laporte, Ecole de Technologie Superieure, Montreal, Canada
Ansik Lee, Texas Instruments, USA
Hareton Leung, Hong Kong Polytechnic University, Hong Kong
Grace Lewis, Software Engineering Institute, USA
Michael Lutz, Rochester Institute of Technology, USA
Andrew Malton, University of Waterloo, Canada
Nikolai Mansurov, KLOCwork Inc., Ottawa, Canada
Esperanza Marcos, Rey Juan Carlos University, Spain
Pat Martin, Florida Institute of Technology, USA
Kenneth L. Modesitt, Indiana University - Purdue University Fort Wayne, USA
Ibrahim Mohamed, Universiti Kebangsaan, Malaysia
James Moore, Mitre Corporation, USA
Keith Paton, Independent consultant, Montreal, Canada
Pedagogy Focus Group Volunteers
Valentina Plekhanova, University of Sunderland, UK
Steve Roach, University of Texas at El Paso, USA
Francois Robert, Ecole de Technologie Superieure, Montreal, Canada
Robert C. Seacord, Software Engineering Institute, USA
Peraphon Sophatsathit, Chulalongkorn University, Thailand
Witold Suryn, Ecole de Technologie Superieure, Montreal, Canada
Sylvie Trudel, Ecole de Technologie Superieure, Montreal, Canada
Hans van Vliet, Vrije Universiteit Amsterdam, Netherlands
Frank H. Young, Rose-Hulman Institute of Technology, USA
Zdzislaw Zurakowski, Institute of Power Systems Automation, Poland

Волонтеры SE2004 по вопросам педагогики
Jonathan Addelston, USA
Donald Bagert, Rose-Hulman Institute of Technology, USA

Jurgen Borstler, Umea Universitet, Sweden
David Budgen, Keele University, United Kingdom
Joe Clifton, University of Wisconsin, Plattsburgh, USA
Kendra Cooper, University of Texas, Dallas, USA
Vladan Devedzic, University of Belgrade, Yugoslavia
Rick Duley, Perth, Western Australia
Garth Glynn, University of Brighton, UK
Elizabeth Hawthorne, Union County College, USA
Orit Hazzan, Technion, Israel
Justo Hidalgo, Universidad Antonio de Nebrija, Spain
M. Umit Karakas, Turkey
Khaled Khan, University of Western Sydney, Australia
Yoshihiro Matsumoto, ASTEM Research Institute of Kyoto, Japan
Pat McGee, Florida Institute of Technology
Andrew McGettrick, University of Strathclyde, USA
Bruce Maxim, University of Michigan, USA
Ken Modesitt, Indiana University - Purdue University Fort Wayne, USA
Steve Roach, University of Texas at El Paso, USA
Anthony Ruocco, Roger Williams University, USA
Peraphon Sophatsathit, Chulalongkorn University, Thailand
Barrie Thompson, University of Sunderland, UK
Yingxu Wang, University of Calgary, Canada
Frank H. Young, Rose-Hulman Institute of Technology, USA

Рецензенты черновых версий SE2004
Robert L. Ashenhurst, Graduate School of Business, University of Chicago, USA
Donald Bagert, Rose-Hulman Institute of Technology, USA
Bruce H. Barnes, USA
Larry Bernstein , Stevens Institute of Technology- Computer Science, USA
Jurgen Borstler, Umea Universitet, Sweden
Vincent Chiew, University of Calgary / Axis Cogni-Solve Ltd., Canada
Tony Cowling, University of Sheffield, UK
Deepak Dahiya, Institute for Integrated Learning in Management, India
Wes Doonan, Movaz Networks Inc., USA
Rick Duley, Murdoch University, Australia
David Parnas, University of Limerick, Ireland
Helen M Edwards, University of Sunderland, UK
Matthias Felleisen, Northeastern University, USA
Maurizio Fenati, Micron Technology Italia, Italy
Robert L. Glass, Computing Trends, USA
Garth Glynn, University of Brighton, UK
William Griswold, University of California, San Diego, USA
Duncan Hall, EDS CPEng, IntPE, MIPENZ; SMIEEE; ACM, New Zealand
Rob Hasker, University of Wisconsin - Platteville, USA
Bill Hefley, Carnegie Mellon University, USA
Jonathan Hodgson, Saint Joseph's University, USA
Vladan Jovanovic, Georgia Southern University, USA

Cem Kaner, Florida Institute of Technology, USA
Pete Knoke, Univ of Alaska, Fairbanks, USA
Hareton Leung, Hong Kong Polytechnic University, China
Tim H. Lin, ECE Department, Cal Poly Pomona, USA
Michael Lutz, Rochester Institute of Technology, USA
Dino Mandrioli, Politecnico di Milano, Italy
Luisa Mich, University of Trento, Italy
Ivan Mistrik, Fraunhofer IPSI, Germany
James Moore, Mitre Corporation, USA
Ana Moreno, Universidad Politecnica de Madrid, Spain
Carl J. Mueller, USA
Ricardo Colomo Palacios, Universidad Carlos III, Spain
Volodymyr Pavlov, eLine Software, Inc., Ukraine
Kai Qian, Southern Polytechnic State University, USA
David Rine, George Mason University, USA
Andrey A.Terekhov, Microsoft, USA
John Walz, Software Quality Consultant, USA
Michael Wing, Vandyke Software, USA
Tony Wasserman, Software Methods and Tools, USA

Участники семинара SE2004 на конференции «Преподавание и тренинги по программной инженерии» (CSEE&T 2002), Ковингтон, штат Кентукки, США, 25 февраля 2002

Группа «Требования»
Keith B Olsen, Utah Valley State College, U.S.
Massood Towhidnejad,Embry-Riddle University, U.S.
Wing Lam, Institute of Systems Science, US
Dennis Frailey, Raytheon Corporation, US
Lynda Thomas, University of Wales, UK

Группа «Проектирование»
Norm Cregger, Central Michigan University, US
Richard Conn, Lockheed Martin, US
John W Fendrich, Bradley University, US
Heikki Saoslamoinen University of Jyvaskyla, Finland
Jim McDonald, Mommouth University, US
David Umphress, Auburn university, US

Группа «Качество»
Elizabeth Hawthorne, ACM TYC & Union County College, US
Michael Ryan, Dublin City University, Ireland
Ellen Walker
Dermot Shinners-Kennedy
Jocelyn Armarego, Murdoch University, US
Ian Newman, Loughborough University, UK

Группа «Процесс»
Orit Hazzan, Technion - Israel Institute of Technology
Jim Kiper, Miami University, US
Cindy Tanur
Dick Lytle, Towson University, US
Rob Hasker, University of Wisconsin - Platteville,US
Peter Henderson, Butler University, US
Nabeel Al-Fayoumi, Ministry of Information & Communication Technology, Jordan
Workshop Organisers
Barrie Thompson University of Sunderland, UK
Helen Edwards University of Sunderland, UK

SE2004 Steering Committee members
Jorge L. Diaz-Herrera, Rochester Institute of Technology, U.S
Thomas B. Hilburn, Embry-Riddle Aeronautical University, U.S.
Rich LeBlanc, Georgia Institute of Technology, U.S.
Ann Sobel, Miami University, U.S.

Участники Международного саммита по преподаванию программной инженерии (SSEE 2002), проводившегося в рамках конференции ICSE 2002, 21 мая 2002:
Paulo Alencar, University of Waterloo, Canada
Joanne Atlee, University of Waterloo, Canada
Pierre Bourque, University of Quebec, Canada
Tony (Anthony) Cowling, University of Sheffield, U.K.
P. Devanbu, University of California, US
Helen M. Edwards, University of Sunderland, UK
Martin Griss, Hewlett-Packard, US
Thomas B. Hilburn, Embry-Riddle University, U.S.
Bob Kossler, University of Utah, US
Timothy C. Lethbridge, University of Ottawa, Canada
Meir (Manny) Lehman, Imperial college, UK
Sam Redwine. James Madison University, US
Karl Reed, Bond University, Australia
Ken Robinson, University of New south Wales, Australia
J. Barrie Thompson, University of Sunderland, UK
Debora Weber-Wulff, Virtvelle Fachhochschule, Germany

Участники семинара SE2004, проводившегося на конференции «Преподавание и тренинги по программной инженерии» (CSEE&T 2003), Мадрид, 20 марта 2003.
Jocelyn Armarego, Murdoch University, Australia
Donald J. Bagert, Rose-Hulman Institute of Technology, USA
Pere Botella, technical University of Catalonia, Barcelona, Spain
Pierre Bourque, University of Quebec, Canada
David Budgen, Keele University, U.K.

John Cooke, University of Sarkatchen, Canada
Tony (Anthony) Cowling, University of Sheffield, U.K.
Jose Javier Dolado, University of the Basque Country, Spain
Gilles Y. Delisle, University of Ottawa, Canada
Robert Dupuis, Universite de Quebec a Monteal, Canada
Helen M. Edwards, University of Sunderland, UK
Jaun Garbajosa, Technical University of Madrid, Spain
Orit Hazzan, Israel Institute of Technology, Israel
Tusto N. Hidaleo, University of Nebrua, Spain
Thomas B. Hilburn, Embry-Riddle University, U.S.
Greg Hislop, Drexel University, U.S.
Rich LeBlanc Georgia Institute of Technology, U.S.
Timothy C. Lethbridge, University of Ottawa, Canada
Keith Mansfield, Addison Wesley/Pearson Education,US
Mike Murphy, Southern Polytechnic State university, US
Faye Navabi, Arazona State University, US
Bill Poole, Seattle University, US
Stephen Seidman, New Jersey Institute of Technology, US
Mark Sebern Milwaukee School of Engineering, US
J. Barrie Thompson, University of Sunderland, UK

Участники Второго международного саммита по преподаванию программной инженерии (SSEE II), проводившегося в рамках конференции ICSE 2003, 5 мая 2003.

James Andrews, University of Western Ontario, Canada
Joanne Atlee, University of Waterloo, Canada
Duncan Clarke University of South Carolina, US
Oliver Creighton, Technische Universitat Munchen, Germany
Geoff Dromey, Griffith University, Australia
Amnon Eden, University of Essex, UK
Helen M. Edwards, University of Sunderland, UK
Keith Frampton, RMIT University, Australia
Keith Gallager, Loyola college, US
Stanislav Jarzabek, National university of Singapore
Cem Kaner, Florida Tech, US
Philip Koopman Carnegie Mellon University, US
Rich LeBlank, Georria Institute of Technology, US
Timothy C. Lethbridge, University of Ottawa, Canada
Pat McGee, Florida Tech, US
James McKim, Rensselaer, US
Everald Mills, Seattle University, US
Hausi Muller, university of Victoria, Canada
Arturo Sanchez, University of North Florida, US
Mary Shaw, Canegie Mellon, US
Andrew Simpson, Oxford University, UK
J. Barrie Thompson, University of Sunderland, UK
Kevin Twitchell, Brigham Young University, US