Что такое программная инженерия

Заведующий кафедрой системного программирования СПбГУ
Генеральный директор ЗАО «Ланит-Терком»
доктор физ-мат наук, профессор А.Н. Терехов

Чуть-чуть истории

Первая [1] ЭВМ ENIAC была создана под руководством Джона Моучли в 1946 году в Пенсильванском университете (г. Филадельфия). В СССР первая ЭВМ МЭСМ была разработана под руководством С.А. Лебедева в Киеве в 1951 году. Правда, в Киеве он проработал меньше двух лет и вернулся в Москву, где в 1953 году в ИТМ и ВТ (Институт точной механики и вычислительной техники АН СССР) создал БЭСМ – самую быструю ЭВМ в Европе того времени.

В США в 1951 году Д.Моучли запустил в серию ЭВМ UNIVAC (было выпущено 47 штук). В СССР первой серийной ЭВМ была БЭСМ-2 (67 штук), которая выпускалась с 1958 года, практически сразу же за БЭСМ-2 была запущена в серию М-20, также серийно выпускались ЭВМ Урал-1 (183 штуки, начиная с 1957 года), с 1960 года – Минск-1 (220 шт.).

В начале 50х годов XX века первые программисты работали один на один с ЭВМ, программируя непосредственно в её кодах. В качестве устройств ввода использовались перфоленты или перфокарты.

Я начал программировать в 1963 году на ЭВМ «Урал-1», где в качестве носителя устройства ввода использовалась кинолента. Мое программирование было не совсем добровольным – мой отец, полковник, инженер по электронике самолетов, уйдя в запас был вынужден переучиваться с аналоговой техники на цифровую. Поскольку он был уже старым, как мне тогда казалось, (тогда ему было 49 лет, а сейчас мне уже 61 – sic transit gloria mundi), он решил, что будет лучше, если сначала я прочитаю книгу Китова и Криницкого [1]) , а уж потом объясню ему. Рука у него была тяжелая, так что возражать было невозможно.

В первые годы программирование занимало малый процент всей работы, нужно было масштабировать (арифметики с плавающей запятой на многих ЭВМ не было), для чего нужно было крутить ручку арифмометра, долго набивать код программы на весьма неудобных устройствах, а вершиной отладочных средств была прокрутка – покомандный интерпретатор.

Еще раз повторю, что первые программисты абсолютно все делали сами. Постепенно пришло понимание, что лучше накапливать библиотеки математических функций, отладочных средств, программ форматного ввода/вывода и т.д. В больших вычислительных центрах появились специальные люди – хранители и толкователи библиотек подпрограмм. В 1957 году появился язык FORTRAN (FORmula TRANslator), созданный под руководством Джона Бэкуса в IBM. Те люди, которые собирали первые библиотеки полезных программ, придумали FORTRAN и реализовали первые трансляторы с него в коды ЭВМ, создали новую науку – системное программирование.

---

[1] Обычно в качестве первой вычислительной машины (если не считать механическую машину Чарльза Бэббиджа – 1837 г.) называют Mark I, созданную в 1943 году в Гарвардском университете под руководством Говарда Эйкена (Howard H. Aiken), но это была релейная, т.е. электромеханическая машина, а первой действительно электронной (на триггерах) стала именно ENIAC.

В СССР системное программирование начало развиваться очень рано. Профессор МГУ А.А. Ляпунов ещё в 1953 году начал работы по операторному методу в программировании, в 1953-54 гг. А.А. Ляпунов читал лекции по программированию будущему академику А.П. Ершову, среди его учеников были уже упомянутые А.И.Китов и Н.А.Криницкий, очень уважаемый мною И.В. Поттосин и многие другие известные программисты. В 1956 году были опубликованы работы Ю.И. Янова по схемам программ, чуть позже работы С.С. Лаврова и А.П. Ершова по экономии памяти на основе раскраски графов. Работы Ю.И. Янова, С.С. Лаврова и А.П. Ершова во всем мире признаны классическими, заложившими основы теоретического программирования.

Системное программирование

Практически все первые теоретики программирования были и великими практиками, они создавали трансляторы, операционные системы, оптимизаторы, шахматные программы-чемпионы мира и т.д. Однако постепенно теория стала отдаляться от практики. Я навсегда запомнил и часто повторяю своим студентам гневную речь А.А. Ляпунова на панельной дискуссии Всесоюзной конференции по программированию в Академгородке: «Не понимаю я современных молодых ученых. В мои годы практика ставила задачи, а теоретики их решали. Часто теория шла впереди практики. А что мы слышали на этой конференции? Оптимизация памяти, но без массивов, корректность программ, но без процедур, goto и присваиваний. Кому это нужно?». Тем не менее, когорта первых системных программистов в СССР была настолько сильна, что практическое направление системного программирования не только не ослабело, но и бурно развивалось.

Таким образом, можно сказать, что первые системные программисты занимались разработкой различных инструментальных средств и новых методов, теорий и алгоритмов, необходимых в этой многотрудной деятельности.

Постепенно системные программисты как специалисты существенно более высокой квалификации, чем обычные прикладные программисты, стали привлекаться к самым трудным задачам – от расчета атомных бомб до управления предприятиями. Иногда это было вынужденной мерой, например, я помню, как трудно шло внедрение первого в СССР транслятора с языка Алгол 68 для ЕС ЭВМ и первых графических технологий. Сначала прикладники (особенно, если это военные люди) «забивают микроскопом гвозди», а потом говорят, что технология плохая. Приходилось засучивать рукава и лезть в самые разные области.

Особого упоминания заслуживают встроенные системы реального времени. Множество параллельно протекающих процессов, особо высокие требования к времени отклика и надежности, непосредственная работа с оборудованием – эти и многие другие особенности систем реального времени создают им ореол самых трудных задач. Я обожаю слушать рассказы А.Н.Томилина о первой в мире системе противоракетной обороны «Система А» (1961 г.), в разработке программного обеспечения для которой он принимал участие. Есть и у меня масса забавных и поучительных историй из собственной практики разработки телефонных станций и различных военных систем. Да-да, именно поучительных. Даже в советские времена, когда никто не спрашивал, хочешь ли ты этим заниматься, мы умудрялись получать бесценный практический опыт, ту самую обратную связь, без которой развитие науки невозможно.

Технология программирования

Термин «технология промышленного программирования» в наш коллектив привнес капитан 1-го ранга профессор В.П. Морозов в конце 1980 года. Поначалу мы были в ужасе, так далеко это было от наших университетских традиций. Я даже специально ездил в Академгородок к А.П.Ершову за советом. Он посмеялся над моими страхами, назвал несколько известных людей, про которых я и не знал, что они генералы или адмиралы. А.П. Ершов поддержал наши начинания, написал развернутый отзыв [2] и даже более двух лет работал у нас академиком-консультантом.

Поначалу такие вещи, как отчуждение программы от её автора, контроль за ходом разработки, графические схемы ситуаций мы называли «шпионскими штучками», но постепенно пришло понимание, что работа коллективов из сотен программистов без таких средств невозможна. Поскольку в западных странах «площадь соприкосновения с пользователем» (выражение А.П. Ершова) у программирования была несравненно больше, там понимание особенностей промышленного программирования пришло несколько раньше. Впервые термин «инженерия программного обеспечения» прозвучал на одноименной конференции НАТО в 1968 году. Я хорошо помню довольно толстую книжку в твердой обложке черного цвета с русским переводом трудов этой конференции [3], но тогда особых откликов в моей душе она не вызвала. Подумаешь, «управление проектами и коллективами» - мы и так все можем! Кстати, на Западе многие ученые тоже восприняли новый термин с прохладцей.

Понадобились годы, чтобы понять, что системное программирование разделилось на более теоретическую часть - информатику (Computer Science) и более практическую программную инженерию (Software Engineering). Сегодня вполне обособленно развиваются также науки, связанные с базами данных, информационным поиском и многие другие.

Программная инженерия

Более 20 лет использование терминов Computer Science и Software Engineering было полностью делом индивидуального вкуса. Например, когда в 1996 году создавалась новая кафедра в СПбГУ, которой я руковожу до сих пор, она по-русски называлась «Кафедра системного программирования», а по-английски – «Software Engineering Chair». Уже тогда я хотел подчеркнуть существенно большую практическую направленность новой кафедры по отношению к «Кафедре математического обеспечения ЭВМ», первым выпускником которой я был. Кстати, именно в момент создания новой кафедры старая кафедра была переименована в «Кафедру информатики» (Computer Science).

С информатикой, т.е. с теоретическим программированием, все было ясно. Уже в 1991 году была опубликована первая международная программа обучения этой науке [4], в 2001 году эта программа была пересмотрена [5]. По крайней мере, все основные определения были зафиксированы, сегодня смешно трактовать информатику не так, как её определяет международный стандарт, все равно, что плевать против ветра.

С программной инженерией всё было сложнее. Многие её направления, особенно связанные с планированием, бюджетированием и управлением воспринимались «в штыки» университетским математическим сообществом. Приходилось буквально прятать эти предметы в спецкурсы с более общими названиями, примерно так же, как это делали генетики ЛГУ за 50 лет до этого. Инженерия не может изучаться в классическом университете! Ей место в технических

вузах! Я прятался за термином «технология программирования» [6], даже как-то раз поплакался Д. Кнуту. Его горячая поддержка мне сильно помогла.

В 2004 году была, наконец, опубликована международная программа обучения программной инженерии [7], в которой, так же как и в программе по информатике, были даны соответствующие определения:

- «Установление и использование правильных инженерных принципов (методов) для экономичного получения надежного и работающего на реальных машинах программного обеспечения» [8].
- «Программная инженерия является такой формой инженерии, которая применяет принципы информатики и математики для получения рентабельных решений в области программного обеспечения».
- «Применение систематического, дисциплинированного, поддающегося количественному определению подхода к разработке, эксплуатации и сопровождению программного обеспечения» [9].

Эти определения можно трактовать в чуть более упрощенном виде: программная инженерия – это наука, которая изучает вопросы создания, сопровождения и внедрения программного обеспечения с заданным качеством, в заданные сроки и в рамках заранее определенного бюджета.

Международные образовательные стандарты

Кажется, основные международные организации IEEE и ACM, имеющие отношение к компьютерам и программированию, всерьез взялись за стандартизацию. В 2005 году был опубликован стандарт Computing [10], включающий в себя:

- Computer Engineering (Разработка компьютеров);
- Computer Science (Информатика);
- Information Systems (Информационные системы);
- Information Technology (Информационные технологии);
- Software Engineering (Программная инженерия).

В этом стандарте достаточно точно и ясно описано, что следует понимать под каждой из 5 перечисленных наук.

Понятно, что «Разработка компьютеров» (стандарт опубликован в 2004 году) ориентирован на тех, кто придумывает и создает новые компьютеры для самых разных применений. В наш век FPGA, VHDL и Verilog этой работой все больше занимаются математики, хотя роль инженеров по-прежнему велика.

Пожалуй, стандарт по информатике наиболее полный и проработанный. В каком-то смысле он является базовым, поскольку на него ссылаются и используют другие стандарты. В 2008 году он был пересмотрен, но, на мой взгляд, изменения не так существенны, как в 2001 году по сравнению с 1991 годом.

Стандарт «Информационные системы» (опубликован в 2002 году, пересмотрен в 2010) чем-то похож на то, что в советские времена мы называли АСУ (автоматические системы управления), но, разумеется, в век Internet, Web 2.0, мобильных устройств, конвергенции телефонов, персональных компьютеров и Internet информационные системы быстро развиваются.

Для меня лично стандарт «информационные технологии» (опубликован в 2006, пересмотрен в 2008 г.) оказался самым загадочным, по крайней мере, под этим термином я понимал нечто другое. Это стандарт для тех, кто строит компьютерные сети, используя маршрутизаторы, прокси-серверы, firewalls и т.д. В России таких специалистов называют системными администраторами, а большие организации – системными интеграторами.

Поскольку программная инженерия является основной для данной статьи, разберем стандарт, опубликованный в 2004 году, подробнее. В стандарте Software Engineering Curricula 2004 [11]рассказана история создания этого документа, основные принципы и цели обучения. Я настоятельно рекомендую своим студентам полистать стандарт, но вовсе не с целью узнать программы отдельных курсов, это, скорее, нужно преподавателям.

На мой взгляд, самая интересная часть стандарта – это короткие эссе, в которых объясняется, что общего и разного между программной инженерией и традиционной инженерией, между программной инженерией и информатикой, между информатикой и математикой. Что-то я не помню в российских образовательных стандартах упоминаний о важности участия в профессиональных сообществах, соблюдения профессиональной этики, умения работать в коллективах, не создавая конфликтов, и т.д. Именно благодаря участию в переводе текста стандарта на русский язык я познакомился к классификацией Блума (знание, понимание, применение), узнал о такой полезной вещи, популярной на Западе, но, к сожалению, не у нас, как SWEBOK [12] – что должен знать и уметь программный инженер через 4 года после окончания вуза.

Итак, программная инженерия включает в себя следующие области знаний:

- Основы компьютинга (основы информатики, технологии и средства разработки, формальные методы);
- Основы математики и инженерии (в том числе инженерная экономика ПО);
- Профессиональная практика (работа в команде, навыки коммуникации, этика);
- Основы моделирования (анализ, работа с требованиями, спецификации);
- Проектирование ПО (концепции и стратегии проектирования, проектирование человеко-машинного интерфейса, средства поддержки проектирования);
- Верификация и аттестация ПО (основы, рецензия кода, тестирование, оценка пользовательского интерфейса, анализ проблем);
- Эволюция ПО (на мой взгляд, совершенно справедливо в стандарте говорится об эволюции, а не о сопровождении ПО);
- Процессы разработки ПО;
- Качество ПО (стандарты качества ПО, процессы обеспечения качества ПО, процесса, продукта);
- Управление программными проектами (концепции менеджмента, планирование и отслеживание выполнения проектов, управление персоналом, управление конфигурацией ПО);

Каждая из перечисленных 10 областей подробно расписана, снабжена примерной оценкой временных затрат, приведены программы соответствующих курсов, часто в нескольких вариантах.

Привязка к российским образовательным стандартам

Стандарт Software Engineering Curricula 2004 разработан очень демократично, предусмотрено множество траекторий обучения, приводятся различные программы отдельных курсов для технических вузов и классических университетов, даны конкретные рекомендации. Приводятся даже примерные учебные планы с разбивкой по годом для разных стран. К сожалению, Россия в эти примеры не попала, поэтому мы с Андреем Тереховым (младшим) взяли на себя труд и риск разработки примерного плана на основе российского образовательного стандарта 01.04 (Информационные технологии). Во всех случаях, когда Software Engineering Curricula 2004 предлагает какой-то выбор, мы на основании консультаций со многими коллегами такой выбор делали, в тех случаях, когда подходящие курсы в 01.04 находились, мы этим пользовались (с необходимыми уточнениями), если же аналогов не было, то добавляли новые курсы (иногда для этого приходилось немного сокращать российский стандарт). Весь этот процесс подробно описан в [13].

Должен сказать, что стандарт 01.04, подготовленный под руководством профессора В.А. Сухомлина (ВМК МГУ), оказался достаточно хорошей основой для выравнивания с международным стандартом. Во-первых, В.А. Сухомлин опирался на Computer Science Curricula 2001, а, во-вторых, в нем было предусмотрено достаточно много часов для курсов по выбору вуза, студента, для самостоятельной работы. Мы этим воспользовались.

В настоящее время (лето 2010 года) накоплен четырехлетний опыт преподавания по этой программе. К сожалению, у нас выдержать план обучения с точностью до семестра не удалось, многие профессиональные курсы были сдвинуты на 1-2 семестра ближе к концу бакалаврской программы. Тому есть две важные причины:

1. Традиционно высокий уровень математической подготовки (что, впрочем, естественно для мат-меха СПбГУ), что потребовало существенно больше часов на младших курсах.
2. Военная кафедра отнимает целый день в неделю на 2-4 курсах. Почему-то в западных университетах такой предмет не предусмотрен.

Итоговая расчасовка нашей программы обучения приведена в Приложении 1. Недавно был принят закон об особом статусе Московского и Санкт-Петербургского университетов, в рамках которого им можно устанавливать свои образовательные стандарты, что мы, естественно, и сделали . Но в этом же законе сказано, что наши университета должны играть ведущую роль при подготовке программ обучения для других университетов России ☺ .

Дополнительное производственное обучение

Стандарт Software Engineering Curricula 2004 достаточно полон и конструктивен. Меня очень порадовало, что в нем много внимания уделяется производственной практике, работе над проектами, подчеркиваются даже важность и преимущества коллективной работы над дипломным проектом.

Мы к подобным принципам пришли более 10 лет назад. Я и сейчас не верю, что можно научить составлению недельных отчетов, управлению конфигурацией ПО, QA (обеспечение качества), работе в команде и т.п. в классе у доски.

Нужно как-то организовывать работу студентов в небольших коллективах над реальными проектами. Но что значит «реальный»? Производственный проект ни один уважающий себя

Заказчик студентам не отдаст, исследовательские проекты требуют большого внимания научного руководителя. Например, наша кафедра отвечает за подготовку примерно 50 студентов на каждом из 3, 4 5 и курсов. Где мне взять несколько десятков квалифицированных научных руководителей для 150 студентов ежегодно?

Поскольку параллельно с преподаванием в Университете я еще руковожу довольно большим предприятием, занимающимся наукоемким бизнесов в области информационных технологий, ответ на поставленный вопрос почти очевиден. Пусть сотрудники предприятия руководят группами студентов. Но зачем им это надо? Всегда на сотню сотрудников найдется 2-3 человека, которым нравится преподавать, даже бесплатно, но вряд ли на этой основе можно наладить регулярную массовую подготовку кадров.

Как обычно, для решения, которое устроило бы многие разные стороны, необходимо совпадение многих интересов и обстоятельств. В данном случае мне на руку сыграло то обстоятельство, что в начале двухтысячных годов рынок кадров в нашей области деятельности был сильно перегрет. Даже весьма слабые специалисты с сильно завышенными требованиями были нарасхват. Еще более трудной представлялась проблема удержания кадров. В Санкт-Петербурге в программистских организациях текучка кадров составляет 15-20%, специалист, которого вы с большим трудом нашли на рынке труда, убежит вприпрыжку, если ему кто-нибудь предложит зарплату на сотню долларов больше.

Таким образом, нужно решать две задачи – нахождение и удержание кадров одновременно [14]. Мы видим это решение следующим образом [15]:

– во втором семестре второго курса мы приглашаем студентов принять участие в 10-12 проектах, каждым из которых руководят не менее двух сотрудников предприятия. Например, в этом году мы предлагали разработку приложений для iPhone и Android, реализацию ОС реального времени, разработку технологии разработки ПО на PHP, CASE-средства на базе библиотеки Qt, разработку нашей версии mindmap (Comapping.com), нескольких игр. Никаких коммерческих результатов не ожидается (но если что-то получится, никто не будет против), организация проектов вполне «взрослая» (планирование, версионный контроль, еженедельные отчеты, QA и т.д.)

– летом после второго курса для лучших студентов мы организуем летнюю школу (1 месяц по 4 часа в день), причем платим им дополнительную стипендию, чтобы как-то компенсировать частичную потерю летнего отпуска.

– начиная с третьего курса мы постепенно, по мере появления вакансий приглашаем студентов на стажерские позиции в предприятии.

Накопленная за 10 лет статистика свидетельствует, что студенты, пришедшие после такой подготовки на работу в наше предприятие, работают у нас много лет, быстро растут профессионально и карьерно. Более того, даже если считать только прямые затраты, оказывается, что находить сотрудников традиционным способом через службу HR обходится существенно дороже затрат на обучение и привлечение студентов. Дело в том, что нового специалиста нужно еще 1,5-2 месяца готовить, вводить в коллектив, отвлекая для этого других сотрудников.

Таким образом, в выигрыше оказываются все:

– студенты, имеющие возможность бесплатно получить дополнительную подготовку, освоить навыки работы в производственном коллективе, изучить новейшие технологии, заранее подыскать себе место работы по душе.

– предприятие получает готовых специалистов, хорошо знающих порядки на предприятии.

– университет получает бесплатно опытных руководителей курсовых и дипломных работ и доступ к новейшим актуальным технологиям.

Заключительные замечания

Программная инженерия – это то, чему мы и наши партнеры учим студентов днем в Университете, а вечером на производстве.

Я всячески приветствую коллективные работы и преемственность курсовых работ с переходом в дипломные работы. С этой целью я подталкиваю сотрудников на масштабные, даже амбициозные студенческие проекты, разумеется, с выделением начальных, более простых этапов. Чаще всего такой подход дает хорошие результаты.

Сегодня наш метод производственного обучения взяли на вооружение Санкт-Петербургские отделения Intel, Yandex, Google, EMC, Exigen и некоторые другие компании, несколько тем предложила компания Транзас, активно работает с нашими студентами IntelliJ. Таким образом, можно сказать, что способ подготовки хороших программных инженеров, разработанный нами, вышел из стадии экспериментов и доказал свои преимущества.

Литература

1. Китов А. И., Криницкий Н. А., Электронные цифровые машины и программирование, изд. второе, Физматгиз, 1961.
2. Как я общался с А.П.Ершовым в формальной и неформальной обстановке, А.Н.Терехов, Сборник трудов конференции «Перспективы систем информатики», Новосибирск, 2010.
3. 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.
4. Computing Curricula 1991. ACM Press, New York, 1991.
5. Computing Curricula 2001: Computer Science, 2001.
6. А.Н.Терехов, Технология программирования, М., ИНТУИТ-БИНОМ, 2007.
7. Software Engineering 2004: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering.
8. Bauer F.L. Software Engineering. Information Processing, 71, 1972.
9. IEEE STD 610.12-1990, IEEE Standard Glossary of Software Engineering Terminology, IEEE Computer Society, 1990.
10. http://www.acm.org/education/education/curric_vols/CC2005-March06Final.pdf
11. Рекомендации по преподаванию программной инженерии и информатики в университетах, М., Интуит, 2007
12. Software Engineering Coordinating Committee. Guide to the Software Engineering Body of Knowledge, IEEE Computer Society, 2001.
13. Computing Curricula: Software Engineering и российское образование, А.Н.Терехов, А.А.Терехов, «Открытые системы», №8, 2006.

14. The Economics of Hiring and Staff Retention for an IT Company in Russia, A.Terekhov, K.Terekhova, Proceedings of 4th International Conference SEAFOOD, 2010.
15. Программа подготовки специалистов в IT-компании, Р.К. Гагарский, Системное программирование, вып.3, Сб.статей, СПб., Издание СПбГУ, 2008.