Преподавание информатики: потерянная дорога

Никлаус Вирт

Приветствие на открытии Международной конференции по преподаванию информатики ITiCSE
г. Аархус (Дания), 24 июня 2002 г.

Всего через пару дней после получения приглашения выступить на открытии данной Конференции по преподаванию информатики, я прочел доклад коллеги из США. Доклад достигает кульминации в следующем абзаце о преподавании:

Поучительно сравнить учебники для средней школы по математике и по информатике. Я имел несчастье проделать это довольно внимательно, и вот мое заключение: мы ни на что не годимся <we suck>. Похоже, мы заставляем студентов сделать вывод, что серьезно думать о карьере в информатике могут только мазохисты.

Мы обязаны поднять свои голоса против традиции, приравнявшей компьютерную грамотность к знанию темных деталей языка программирования, используемого в индустрии.

И я с этим согласен. Посвятив существенную часть своей карьеры доведению искусства создания программ до такого уровня, чтобы его можно было преподавать методично и систематически, я разочарован в доминирующих разрушительных тенденциях. Хоть я и устал от непопулярной роли вечного критика, процитированная статья вновь всколыхнула эмоции, и вот я здесь, поскольку упомянутый доклад продолжается так:

Как профессионалы в информатике, мы обязаны поднять свои голоса против традиции, приравнявшей компьютерную грамотность к знанию темных деталей языка программирования, используемого в индустрии.

Вспоминается рассказ Э.Дейкстры о его ночном кошмаре после чтения спецификаций нового языка программирования PL/1 в 1965 г. Ему представилось, что в будущем *программирование*приравняют к *выучиванию PL/1,* а *информатику* — к *овладению OS/360 JCL* <речь идет о языке программирования и языке управления заданиями для компьютеров фирмы IBM, печально известных своим крайне неудачным дизайном; российские программисты старшего поколения помнят, что это такое, по опыту работ на ЕС ЭВМ — *прим. перев.*>. Достаточно заменить PL/1 на C++ или Java, а JCL — на Windows или Linux, и вы чудесным образом перенесетесь в настоящее.

Тогда я написал своему коллеге о полном согласии с ним. Он ответил следующим разъяснением:

Мои резкие замечания о преподавании — результат полного провала попыток помочь сыну, ученику старших классов, освоить C++. Дизайн языка чудовищен, а учебник написан отвратительно. Мой сын не мог понять, почему x = y должно отличаться от y = x. Дейкстра жаловался мне, что важная книга по языку Java не содержала формальной грамматики.

Действительно, формальные правила синтаксиса были заданы лишь в четвертой версии языка! Но позвольте мне продолжить цитату:

Я был разочарован не только таким положением вещей, но и тем, что серьезные специалисты по информатике воспринимают его как совершенно нормальное. Мне еще ни разу не попадался учебник по UNIX/C++/Java, который я мог бы освоить за неделю. Их учебники невозможно читать, они предполагают, что читатель принадлежит какой-то секте, чьи заклинания должны оставаться тайной для публики, и читателю не следует ожидать многого в плане надежности, связности или общей элегантности. Мое отчаяние достигло апогея, когда я пытался научить своего сына программировать на C++ — факультативный курс в средней школе! После полугода агонии — как для отца, так и для сына — я посоветовал сыну бросить этот курс.

Чего я не понимаю, так это отсутствия возмущения среди ученых, специалистов по информатике. Когда управляющий совет колледжа решил включить в программу C++ в середине 90-х гг, я письменно выразил им свое возмущение. Но я не в счет.

Преподавание в средней школе оказывается отличной проверкой не только способности к преподаванию, но и ясности учебников. Время от времени я получаю жалобы от учителей, сообщающих о трудностях, которые они испытывают с современными средствами и языками программирования (и нередко они просят версию старого Паскаля для новых машин). Вот выдержка из переписки с физиком-теоретиком из России, который предпринял эксперимент по преподаванию программирования способным ученикам старших классов лицея. Он пишет:

Интересно, что курс в лицее помогает лучше понять как само программирование, так и то, как его нужно преподавать. Примерно половина улучшений в моем университетском курсе будет сделана благодаря опыту, полученному в лицее. Кроме того, я почувствовал, в каком ужасном состоянии находится преподавание программирования — как на уровне средней школы, так и университета.

Посмотрим правде в глаза: разве большинство

Слова резкие, но они не преувеличение. В чем же ошибка? Что можно сделать? Вспоминается смиренное “Я не в счет” из процитированного выше письма. Мы чувствуем себя беспомощными. Кое-кто чувствует себя обреченным на жалобы, а большинство решают примириться с очевидными фактами и кое-как приспособиться к ним. Но вряд ли подобную позицию интеллектуальных лидеров можно оправдать. Посмотрим правде в глаза: разве большинство учреждений образования не оказалось заложниками горстки компаний, чья профессиональная цель состоит в повышении доходов — идет ли речь о производителях оборудования, программного обеспечения или об издательствах? Университеты, которые могли бы оказывать хоть какое-то корректирующее влияние, и чьи преподаватели числятся среди авторов популярных учебников, на самом деле больше заинтересованы в том, чтобы демонстрировать свое участие в модных исследованиях, оставляя преподавание ассистентам.

учреждений образования не оказалось заложниками горстки компаний, чья профессиональная цель состоит в повышении доходов?

Конечно, в такой постмодернистской академической среде профессор давно перестал быть мудрецом, углубляющимся все дальше в свой излюбленный предмет в тиши кабинета. Современный профессор — это менеджер большой команды исследователей, хваткий добытчик грантов, поддерживающий тесные связи с ключевыми организациями-источниками финансирования, и неутомимый автор волнующих проектных заявок и впечатляющих отчетов о достигнутых успехах. В этом высоко конкурентном бизнесе было бы самоубийством растрачивать время на размышления о том, как лучше рассказать о простых вещах массе начинающих. Когда речь заходит о материалах для курса, программном обеспечении и т.п., очевидный выбор — взять то, что лежит на полке и заведомо принято всеми остальными. В этой борьбе за успех и выживание лучше всего примкнуть к толпе. Достижения измеряются размером команды, количеством публикаций, цитирований и докладов на конференциях и использованными ресурсами — но не преданностью делу преподавания, которую все равно невозможно измерить. Разумеется, такой стиль академической жизни нередко противоречит внутренним убеждениям индивидуума, но навязывается давлением извне превратить храмы учености в хорошо разрекламированные источники доходов, и этот стиль граничит с проституцией.

Слова резкие. Но что же можно сделать? Ведь и в самом деле индивидууму очень трудно противостоять глобальной тенденции. Я имею в виду тенденцию перехода от долгосрочного планирования к немедленному извлечению доходов, от учения с целью понимания к применению навыков

Конечной целью образования должно быть искусство конструктивного мышления. Именно в таком контексте становится важным наличие хорошо спроектированного языка программирования.

непосредственного действия. Что касается нашего предмета — информатики и программирования для компьютеров, — то конечная цель учреждения образования должна быть гораздо шире, чем овладение каким-либо языком программирования. Это должно быть никак не менее, чем искусство проектирования артефактов для решения сложных задач. Иногда это называют искусством конструктивного мышления. Именно в таком контексте становится важным наличие подходящего инструмента, хорошо спроектированного языка программирования. Он играет роль теории, на которой основываются наши методы. Как можно научиться хорошему и эффективному проектированию, если базовый формализм, само основание представляет собой ошеломляющую, непостижимую путаницу? Как можно освоить такое искусство без образцовых примеров, достойных изучения и подражания? Конечно, не все равно одарены в том, что касается хорошего проектирования, и все же надлежащее обучение, инструменты и примеры играют важнейшую роль.

Я говорил о проектировании сложных артефактов. Очевиден факт, что наши артефакты становятся все сложнее. Например, автомобиль это уже не просто двигатель и четыре колеса. В нем еще есть компьютер для определения оптимального количества впрыскиваемого топлива в самые подходящие моменты времени. Это нужно, чтобы снизить затраты топлива, чтобы сберечь энергию. Но современный автомобиль еще содержит с десяток (или больше) хорошо спрятанных электромоторов для управления окнами и антеннами, радар для определения расстояния до опасных объектов, системную шину для связи всех этих устройств, а также компьютер для управления ими. Эти вещи не слишком способствуют истинному назначению автомобиля, но они добавляют сложность, затрудняют обслуживание и повышают стоимость. Это мнение было выражено в недавней статье о "совершеннейшем автомобиле" в газете Нью-Йорк Таймс (озаглавленной *«За рулем, BMW 745i, технический нокаут»*):

Люди по ошибке принимают сложность за изощренность.

«Люди по ошибке принимают сложность за изощренность,» сказал однажды Никлаус Вирт, швейцарский ученый, специалист по информатике. В роскошных автомобилях, напичканных новейшими технологиями, прибамбасами и завлекалками, уже трудно увидеть эту разницу — так же, как трудно определить ту точку, за которой техника, целью которой была помощь водителю, становится помехой вождению … Достаточно сказать, что семерка, несомненно, самый «продвинутый» седан в мире, но отнюдь не самый легкий в управлении.

Ненужная, самодельная сложность порождает множество проблем.

Этот пример легко переносится в область компьютерных и программных систем. Они стали безмерно сложными не столько потому, что все их чудесные средства и возможности были на самом деле нужны, сколько просто потому, что они были возможны. Поэтому производители надеются, что они дадут преимущество перед конкурентами.

«Преимущество», однако, приводит к избыточной громоздкости, трудности использования и снижению надежности. Но обманутый клиент понимает это только после покупки.

Вывод состоит в том, что ненужная, самодельная сложность порождает множество проблем. Главное, она размывает различием между тем, что действительно важно (оптимальное впрыскивание топлива), и тем, что эфемерно (моторчики для закрытия окон). Беда в том, что требуется гораздо больше таланта, проницательности и времени, чтобы спроектировать экономную, простую и эффективную систему, нежели сложную и громоздкую.

Требуется гораздо больше таланта, проницательности и времени, чтобы спроектировать экономную, простую и эффективную систему, нежели сложную и громоздкую.

Хороший дизайн должен быть в центре нашего преподавания.

Итак, хороший дизайн должен быть в центре нашего преподавания. Но как нам учить образцовому дизайну с помощью инструментов и языков, которые делают нас посмешищем? К нашему сожалению, индустрия программирования сделала немного, чтобы помочь нам, преподавателям, преодолеть наши трудности.

Давайте поэтому взглянем на индустрию программирования, которая оказывается неспособной обеспечить нас идеальными инструментами. Мы увидим, что она сама страдает от чрезмерной, немотивированной сложности, а также от отсутствия регулярности и надежности в своих продуктах. На самом деле индустрия была бы гораздо производительней, если бы строила свои системы на прочной основе вместо того, чтобы прививать новые ростки на гниющие стебли. Мне известно о случае, когда в большой компании был запущен проект, чтобы с нуля спроектировать замену широко используемому программному приложению, ставшему слишком громоздким и трудным в сопровождении. Однако через некоторое время проект был закрыт по совету отдела маркетинга. Решение было принято из-за всеобщего сопротивления клиентов любому изменению, а следовательно и усовершенствованию. Среди клиентов было много и учреждений образования. Им нужно было сохранить свои инвестиции в создание курсов и курсового материала.

Кроме того, вузы становятся все более похожими на коммерческие предприятия, предлагая то, что требуют и за что платят их клиенты, вместо того, что более способствовало бы развитию в долгосрочной перспективе. Но студенты сосредоточивают свое внимание на том, что даст им лучшие шансы в поиске работы, т.е. на овладении навыками, необходимыми в данный момент для работы на предприятиях индустрии.

Очевидно, перед нами в высшей степени устойчивый порочный круг: учителя не могут изменить свои курсы, т.к. они должны привлечь и доставить удовольствие студентам; студенты требуют то, что

практикуется в промышленности; а индустрия применяет и воспроизводит то, чему обучены ее работники. Этот замкнутый круг напоминает ситуацию, описанную мной во введении к сообщению о Паскале в 1970 г.: вузы стремятся учить тому, что требует индустрия, а в индустрии практикуется то, чему ее работники выучились в вузах.

Программирование является, возможно, самой важной новой дисциплиной постиндустриальной эры.

Порочные круги существуют, чтобы их разрывать. Делать это должны те, кто обнаруживает их порочность. Беда сегодня в том, что эта долгосрочная порочность недостаточно признана. Однако программирование, как искусство конструктивного дизайна, слишком важно, чтобы пожертвовать им в пользу краткосрочных коммерческих выгод и привычек. Программирование является, возможно, самой важной новой дисциплиной постиндустриальной эры.

Часто обвиняют могучую индустрию программирования в навязывании своих продуктов беспомощному сообществу клиентов, включая пренебрежимое меньшинство преподавателей и студентов. Однако ситуация еще хуже: преподаватели большей частью добровольно поддались доминирующим коммерческим тенденциям, зачастую ощущая при этом угрызения совести и открыто жалуясь на свои горести, имеющие причиной их собственную позицию. Многие в индустрии сожалеют об этом отречении от лидерства и ответственности.

Только университетские преподаватели в состоянии сломать этот порочный круг ... Они просто обязаны подняться до роли лидеров.

Только университетские преподаватели в состоянии сломать этот порочный круг. Это сделать нельзя ни быстро, ни легко. Но если это окажется невозможным, то что-то, видимо, глубоко неправильно с преподавателями и их академической свободой. Они просто обязаны подняться до роли лидеров.

Порочный круг был однажды разорван, когда распространился Паскаль. При поддержке коллег-единомышленников и в упорном противостоянии рутинерам, Паскаль распространился в учебных заведениях и проник в индустрию. Это произошло, несмотря на могучую конкуренцию со стороны индустрии и других больших организаций, в соперничестве с языками PL/1, Алгол 68 и Ада. Однако наследники Паскаля, существенно его превосходившие, Модула-2 и Оберон, не получили должного внимания среди преподавателей, и сами пали перед лицом самого недостойного из соперников — С. Самого недостойного, т.к. в этом языке были нарушены все открытые к тому времени принципы серьезного программирования. Он запутывает студентов, допуская разный смысл для x = y и y = x и принуждая всех писать x = = y вместо обычного x = y. Только за одни эти пороки он заслуживает изгнания из учреждений образования. Однако сей уродливый синтаксис был целиком

воспроизведен в языке Java, принятие которого академическим сообществом произошло, отчасти благодаря этой преемственности.

В ряде университетов на общепризнанный разрыв между желательным, с точки зрения образования, и практикой "реального мира" был дан ответ посредством выбора функционального языка для первоначального обучения программированию. Однако эта увертка только увеличила разрыв, т.к. теперь обучение стало вестись в рамках другой парадигмы программирования и мышления при создании программ и, как следствие, требуется известное усилие при переходе от обучения к профессиональной деятельности. В результате научная дисциплина и инженерная практика программирования стали восприниматься как разные сущности, почти не имеющие видимой связи. Первая осталась своего рода искусством для искусства, вторая эволюционирует как комбинация эвристик и интуиции, все более изощренного hacking'а. Но все это не может служить фундаментом научной дисциплины, каковая должна, в конце концов, лечь в основу любых инженерных конструкций.

И все же борьба с порочным кругом не совсем безнадежна. Мы указали на тех, кто должен возглавить атаку. Но как?

... я вижу в своем воображении образцовый учебник в качестве подходящего исходного пункта.

Позвольте мне закончить выступление смелым предложением для этой просвещенной аудитории профессионалов преподавания. Я вижу в своем воображении образцовый учебник в качестве подходящего исходного пункта. Он должен удовлетворять следующим критериям:

1. Начинаться сжатым введением в основные понятия программного проектирования.
2. Использовать лаконичную формальную нотацию, строго определенную не более чем на примерно 20 страницах.
3. Основываясь на этой нотации, вводятся основные понятия итерации, рекурсии, логического утверждения <assertion> и инварианта.
4. Центральная тема — структурирование утверждений и типизация данных.
5. За этим следуют концепции упрятывания информации, модульности и проектирование интерфейсов, продемонстрированные образцовыми примерами.
6. Книга устанавливает терминологию, которая столь же интуитивна, сколь и точна.
7. Книга имеет умеренный размер.

Позвольте мне заключить еще двумя замечаниями. Мой коллега, чьи слова приведены в начале доклада, закончил так:

Руководящим для моей карьеры в преподавании и исследованиях был тот принцип, что хорошо подготовленные профессионалы должны быть

гораздо эффективнее, чем вдохновенные любители. В их производительности должно быть различие, и притом существенное. Думаю, что нашей общей целью должно быть увеличение этого различия.

Несколько месяцев назад я получил просьбу дать список задач, которые я считаю первостепенными для информатики на ближайшие десятилетия. Возможно, создание подобного учебника следовало бы включить в этот список, и может быть даже первым номером. Во всяком случае, эта задача пока не решена.

Приветствие на открытии Конференции ITiCSE, Аархус (Дания), 24. 6. 2002