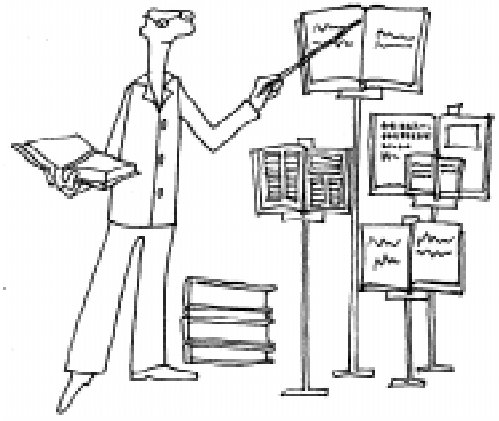
Терехов Андрей Николаевич

КАК ГОТОВИТЬ СИСТЕМНЫХ ПРОГРАММИСТОВ

С каждым годом растет интерес выпускников средней школы к профессии программиста. Профессия программиста привлекает еще и потому, что считается высоко оплачиваемой. В этом году многие выпускники математических школ, мечтавшие стать «чистыми математиками», выбрали для себя профессию программиста. Тем не менее, далеко не все стремящиеся стать программистами представляют, что нужно знать тем, кто хочет стать высококвалифицированным специалистом в области программирования. Многие ребята считают, что знание одного или нескольких языков программирования уже делает их программистами. Редакция журнала обратилась к одному из ведущих специалистов в области программирования, являющемуся одновременно крупным ученым и руководителем созданной им же фирмы, заведующему кафедрой системного программирования Санкт-Петербургского государственного университета проф. А.Н. Терехову с просьбой высказать свое мнение о подготовке системных программистов вообще и в СПбГУ, в частности.

Профессиональным преподавателем Университета я стал почти случайно. Я читал спецкурсы, будучи еще студентом математико-механического факультета, руководил дипломными работами, 9 человек защитили кандидатские диссертации под моим руководством. Еще в молодые годы я начал руководить лабораторией системного программирования НИИ математики и механики математико–механического факультета. Когда уехал работать во Францию заведующий кафедрой математического обеспечения А.О. Слисенко (сейчас он заведует кафедрой в университете Париж-12), наш декан решил, что я буду хорошей кандидатурой на этот пост. На собрании кафедры меня попросили рассказать о своей программе. Она была очень короткой, всего из двух пунктов. Первый тезис: каждый преподаватель должен быть сначала исследователем, а уж потом преподавателем. Я готов простить некоторые недоработки, но не готов простить начетничества, когда преподаватель сегодня почитает книжку, а завтра расскажет. Надо, чтобы преподаватели в основном рассказывали о своих работах или о тех, в которых они принимали участие. Второй тезис состоял в том, что надо соответствовать международным программам.

За каждым из этих тезисов была моя выстраданная позиция. Не люблю я «на-

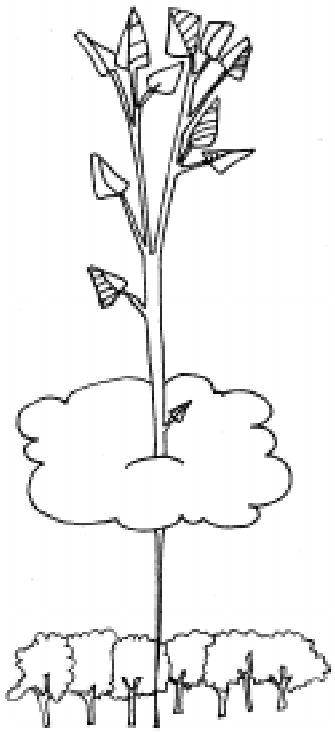


Не люблю я «начетчиков»...

четчиков». Страдал от таких преподавателей, когда сам учился, и, естественно, не хочу поддерживать их сейчас. А с международными стандартами бывали ужасные истории. Например, мы по роду своей деятельности много контактировали с группой молодых людей из Академгородка из Новосибирска, которые под руководством доктора наук Котова делали новую машину «Кронос». Мы в это время делали свою машину «Самсон», поэтому очень интересно было поговорить, пообщаться, обменяться результатами, и ребята произвели на меня незабываемое впечатление. Они часто приезжали к нам, жили у меня дома. Многих из них я хорошо помню до сих пор. Двое из них в конце 80-х годов пробовали поступить в аспирантуру в американском университете. Оба были очень умными, я мечтал бы иметь таких сотрудников. И не поступили. Не потому, что плохо говорят по-английски, или по какой-то еще формальной причине. Они просто на половину вопросов не знали ответов. Хотя у них была очень мощная поддержка. Руководителем их лаборатории был А. Марчук, а его отец был президентом Академии наук, поэтому сотрудники А. Марчука имели дополнительные возможности, получали доступ к материалам, связанным с аспирантурой в Америке, которых младшие научные сотрудники других организаций не имели.

На меня это произвело оглушающее впечатление, потому что я привык думать, что мы, по крайней мере, в области программирования «впереди планеты всей». В некоторых областях это действительно так. В области техники трансляции, в области теоретических вопросов программирования, теории оптимизации. Но оказалось, что программирование за это время разрослось, и мы в своих работах, в основном, на оборону, очень многие аспекты просто упустили. И в 1992 году, по моим подсчетам, мы не охватывали даже 40% международного стандарта по специальности computer science and software engineering. Я сказал сотрудникам кафедры, что нечего почивать на лаврах, нужно засучить рукава и заниматься, догонять мировую цивилизацию. Была масса проблем, были дискуссии на Ученом Совете. Были неприятные и даже болезненные ситуации, но в результате сформировалась новая кафедра.

Так 6 лет назад я стал заведующим кафедрой системного программирования. Я начал честно воплощать собственную программу, развивать исследования, которых у нас раньше не было. Думаю, что сейчас мы охватываем примерно 80% международного стандарта, но не могу обещать, что скоро мы охватим все 100%.



...кафедра системного программирования, несмотря на свою молодость, развивается достаточно успешно...

Именно сотрудники нашей кафедры руководят командами нашего Университета на международных соревнованиях. Мы дважды стали чемпионами, но для меня еще важнее, что в течение 5 лет подряд мы были в призовой «десятке» из 2500 команд. Заметна стабильность результата. Я считаю, что кафедра системного программирования, несмотря на свою молодость, развивается достаточно успешно.

Сейчас я хочу сосредоточиться не на успехах (я отчетливо понимаю – сегодня есть успех, а завтра тебя никто не вспомнит), а на проблемах, которые мешают нам развиваться дальше. Их несколько, и я не знаю,

сумею ли я связно о них рассказать в этом интервью. Но попробую.

Начну я, как ни странно это, возможно, со стороны покажется, с практики. Студенты должны иметь практику. Программирование – это такая специальность, которой не научишь у доски с мелом в руках. Для того, чтобы лучше понять возможные пути организации практики, мысленно перенесемся в Оксфордский университет, где мне доводилось читать лекции, и я специально изучал местную постановку образования. Конечно, там иногда отдает некоторой «замшелостью», но, тем не менее, сотрудники университета свято следуют традиции и с большой неохотой расстаются с чем-то старым. Иногда им это можно поставить в минус. Например, сейчас Оксфордский университет несколько отстал в области естественных наук от Кембриджа, и специалисты говорят, что одна из причин этого отставания в том, что в Оксфорде на 60 лет позже отменили обязательное обучение латыни. В Кембридже отменили в 20-х годах, а в Оксфорде только в 80-х. И эти 60 лет многих молодых людей отпугивала необходимость учить мертвый язык только потому, что так делали 800 лет назад.

Верность традициям достойна уважения. Например, в Оксфорде для каждого предмета есть теоретический курс, то есть лекции, практический курс, когда преподаватель со студентами решает задачи в аудитории, у доски, у компьютера, но при этом стоит рядом, и практикум – по каждому курсу студент должен выполнить некоторую работу самостоятельно. Причем не раз в полгода, как наша курсовая работа, одна на весь семестр, а по каждому предмету каждые две недели. Есть огромные аудитории, сотни свободных вычислительных машин. Приходишь, садишься, решаешь, и затем свой результат показываешь тьютору. Это еще одна особенность – персональное обучение. По каждому предмету час в неделю студент работает со своим тьютором – таким преподавателем, который за него отвечает. Было пять предметов в неделю – значит, пять часов студент с преподавателями проведет один на один. Причем тьютор составляет основу преподавания. Именно тьютор ведет вступительные экзамены, именно тьютор отбирает себе студентов, а не колледж или университет в целом. Именно тьютор может сказать своему коллеге: «Знаешь, я уже набрал себе нужное количество студентов, есть еще такой-то студент, возьми его себе в другой колледж».

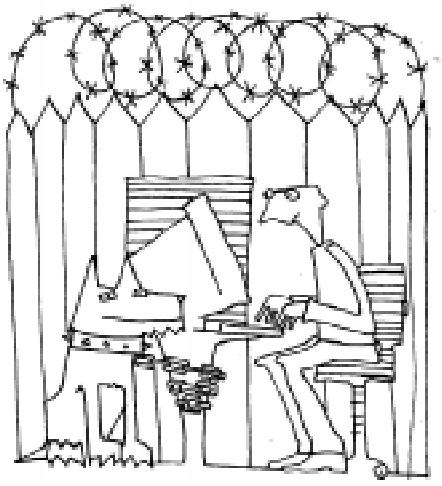
Если читают, например, операционную систему реального времени, то каждый студент должен написать программу: управление памятью, управление процессорами, управление временем. Преподаватель смотрит не только на результат, но и на то, как написана программа. По всем предметам есть лекции, практика и практикум.

У нас с этим слабее. Курсовые работы – раз в семестр и часто превращаются в фикцию. Трудно изменить эту ситуацию к лучшему, потому что нет соответствующих материальных ресурсов. Как обязать всех студентов по каждому предмету сделать самостоятельную работу, если мы не можем им обеспечить полноценный доступ к вычислительным машинам? Классы всегда перегружены. У нас не принято, чтобы студенты, занимались без преподавателя: и вирус занесут, и что-нибудь украдут, и что-нибудь сломают. Самостоятельную работу очень трудно наладить. О тьюторстве я даже не мечтаю. Это прежде всего вопрос денег. У нас наверняка нашлось бы много хороших преподавателей, но для того, чтобы обеспечить индивидуальное обучение, сколько надо преподавателей и какое потребуется финансирование? Хотя еще со средних веков известно, что обучение – это всегда работа мастера с подмастерьем с непосредственной передачей опыта. Единственная «живая» практика у нас – на пятом курсе, полугодовая преддипломная практика. И здесь тоже есть свои проблемы. Хорошо, если практика была в известной

фирме, которая успешно и продуктивно на современном технологическом уровне занимается программированием. Но так бывает не всегда.

Отсюда первый тезис – сегодня надо практику реально совместить с теорией. Формально говоря, все у нас есть. Студенты пятого курса полгода проходят преддипломную практику. И чем это кончается? Вот у меня толстая пачка отчетов по практике. Люди просто пристраиваются работать (например, программистами) в какую-нибудь малоизвестную контору, чаще всего я даже названия этой конторы не знаю. Они работают. С одной стороны, какие могут быть претензии? Человек полгода проработал, получал деньги, кому-то был полезен. Спрашиваю я как заведующий кафедрой: «Чему ты там научился?». «Вот, получил практический опыт программирования на Java или C++». «Как была организована работа?». «Никак. Начальник дал задание, я написал программу». «Как был организован коллектив? Какие были взаимоотношения? Как велось планирование, отчетность? Были ли еженедельные собрания? Была ли регулярная проверка качества? Были ли перекрестные чтения?». «Не было». «Так чему ты, милый, там научился?». «Программированию». «Почему тогда надо говорить, что ты заканчиваешь математико-механический факультет старейшего университета России?». Здесь явное несоответствие теории и практики.

Приведу другой пример. У нас несколько человек проходили практику в зарубежной фирме здесь, в Санкт-Петербурге. В этой фирме все хорошо организовано, но есть другая крайность: везде завеса секретности. Даже если дипломную работу студент написал там, ее в университете защитить нельзя. Надо защищать в фирме, организовывать ГЭК. Это целая проблема. Ладно, в конце концов, даже на военных работали, могли все организовать. Но ведь человек отрывается от коллектива, ничего не может обсудить. Самое главное в обучении – это беседа, разговор. А тут несколько человек проходят практику и даже со своими однокурсниками не общаются. Запрещено. Совершенно другая крайность. Очень высокий уровень работы, но слишком индивидуальный. Люди из этой фирмы, возможно, скажут, что я не прав, что у них есть и семинары и регулярное обучение. Но я говорю о том, что вижу по результатам работы наших студентов.



...везде завеса секретности...

Некоторые студенты проходят практику на предприятиях. Я требую, чтобы это была не только работа, но и обучение. Тоже возникают противоречия: «Мы производственное предприятие, нам надо зарабатывать деньги, приносить прибыль, поэтому заниматься чисто учебными делами как-то не с руки. Нет-нет, мы понимаем, что надо готовить кадры, но все должны заниматься своим делом».

Еще раз повторю, поскольку для меня это важно: практика – это не просто работа «от забора до обеда», переделал готовую программу или спаял электронную схему. Практика подразумевает некоторое исследование, обучение организационным формам, современным методам. Практика должна быть разнообразной. Например, электронщик должен не только спаять, но и спроектировать схему, спроектировать кристаллы, которые внутри. Он должен участвовать в отладке и не просто участвовать, а сделать тесто-

вое окружение. Электронщики должны программировать. Это все тесно связано.

Вас не удивляет, что я все время вспоминаю электронику? У меня есть целый отдел электронщиков, у них и руководитель отдела – математик, и многие сотрудники – математики. Сейчас электроника такая, что все равно надо программировать. Но приходят инженеры, которые не знают, как простейший тест написать. А как написать не один тест, а систему тестов для исчерпывающего тестирования, им даже объяснить невозможно. Практика должна включать в себя организационные аспекты, элементы дизайна, элементы разработки и, самое главное, – доводку до результата. За полгода всегда можно получить результат. На эту тему можно было бы еще о многом сказать.

Тезис второй – чему учим? Вот передо мной лежит программа 35.15. По этой программе учится отделение информатики математико-механического факультета. Мы с сотрудниками нашей кафедры принимали участие в ее разработке. Для сравнения скажу: у нас отделение прикладной математики учится по специальности 01.02. Математическая статистика, моделирование, теоретическая кибернетика – это все замечательно. В дипломе написано: «Математик. Системный программист». Я обращаюсь к авторам этой программы: «Покажите, где здесь программирование». На первом курсе учат программированию на языке С, и все. Я же не говорю, что в программу включили лишний материал. Все нужные вещи: и моделирование нужно, и кибернетика нужна, и распознавание образов, и вопросы оптимизации. Но зачем пишут в дипломе «Системный программист»? Вот я заведую кафедрой системного программирования. Надеюсь, что я знаю, что такое системное программирование. Давайте я тоже буду учить программированию, а писать в дипломе «специалист по методу Монте-Карло». Кому это понравится? Конечно, все понимают, что надо привлечь людей, звучит название специальности хорошо, но не совсем соответствует содержанию курса.

Вернемся к тому, что я действительно считаю хорошим. Например, к специализации 35.15: математические основы информатики, информационные системы, технологии программирования, архитектуры вычислительных систем, сети. Далее: экспертные системы, теория оптимизации баз данных, интернет и интранет, инструментальные системы для C++, Java-технологии, инструментальные средства визуального программирования, инструментальные средства логического программирования, технология трансляции, языки и системы программирования, архитектура ЭВМ, программно-аппаратные комплексы, операционные системы реального времени, телекоммуникации и так далее.

Мы на кафедре подсчитали часы по этой программе, все равно 50 % – это «чистая математика». Самый главный недостаток даже не в этом, а в том, что та половина времени, которая отведена на специальность, отнесена на конец. На первых трех семестрах – только 4 часа в неделю. Представляете себе, человек поступил на отделение информатики, не на отделение «чистой математики», не на отделение астрономии или механики. И учится полтора года, три семестра, имея 4 часа программирования в неделю! В самых разных видах, все про все. Как можно его научить? Самое ценное время уходит. Я даже встречался с заместителем министра образования, обсуждал это все у нас в Университете, в УМО. Сценарий разговора всегда был таким: «На кого жалуетесь, вы же сами профессор, член УМО! Вот и вносите предложение, сократите то, добавьте это, для чего и создано УМО». Хорошо. Когда я только пытаюсь это делать, мне сразу говорят: «Как? Ты что? На факультете работают старые профессора, которые и тебя учили математическому анализу, алгебре, высшей геометрии... Если ты уменьшишь нагрузку, их надо будет сокращать. Неужели ты хочешь уволить старых профессоров-математи-

ков?» Конечно, не хочу. Правила, установленные Министерством, предписывают, что количество преподавателей должно быть связано с количеством студентов. Если число студентов уменьшилось, соответственно уменьшается число преподавателей. То есть такая простая вещь, как перераспределение часов, сталкивается с министерскими правилами, и никакой Университет, никакое УМО изменить это не может.

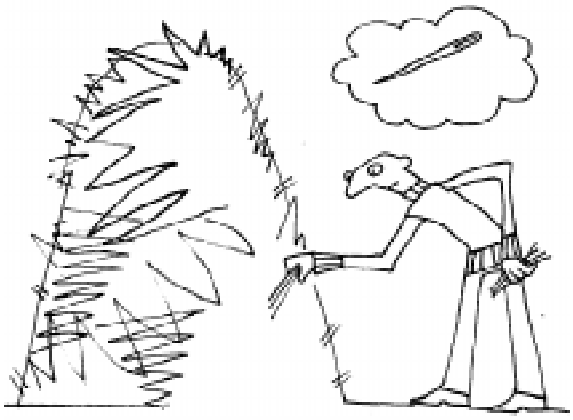
В нашей программе есть федеральный компонент, вузовский компонент (региональный) и по выбору. Федеральный компонент: математический анализ – 4 семестра, количество часов в неделю: 8, 4, 6, 6. Алгебра и теория чисел – 3 семестра, часы: 4, 4, 4. Геометрия и топология – 3 семестра, часы: 4, 4, 4. Дифференциальные уравнения – 2 семестра, по 4 часа. Функциональный анализ один семестр, 4 часа. Когда я был студентом, было два семестра. Уравнения математической физики – один семестр, но 6 часов. Теория вероятностей и математическая статистика два семестра, 7 часов.

Когда я учился, вся теория вероятностей ограничивалась изучением меры Лебега. О том, что вероятность находит применение в нашей науке, я узнал лет через 15: оказывается, отказы вычислительной техники распределены по закону Пуассона. Так можно оценить вероятность отказа, но узнал я об этом только когда столкнулся на практике. Мы сделали новую вычислительную машину, от нас потребовали расчет, я взялся за книги и с удивлением узнал, что теория вероятностей – полезная наука. Мне было уже под сорок. Ничему такому, как предсказывать отказы, как считать их интенсивность, – ничему этому нас не учили. Одни интегралы, интегралы, интегралы. Нет, из этих интегралов потом следуют и закон Пуассона, и все остальное, но мостика между мерой Лебега и еще чем-нибудь полезным нет.

Есть вычислительный практикум – три семестра по 2 часа, и есть программирование – три семестра, 3, 2, 2 часа. А здесь должны быть общепрофессиональные дисциплины (федеральный компонент): архитектура вычислительных систем компьютерной сети, операционная система оболочки, структура алгоритма компьютерной обработки данных, базы данных и СУБД, компьютерное моделирование, компьютерная графика, теория формальных языков и трансляций, спецкурсы по выбору, спецсеминары. Но против всего этого – пустые клетки.

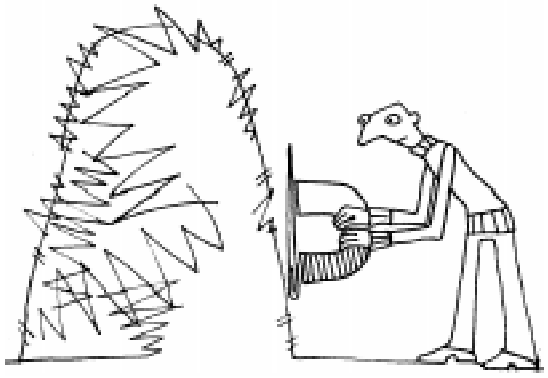
Посмотрим на третий курс (пятый-шестой семестр). Десять часов в неделю. Как можно научить студентов? И только на 4–5 курсе начинают учить «по специальности». Но на пятом курсе уже преддипломная практика, там только спецкурсы, и то понемножку. То есть мы можем учить практически только четвертый курс. Разве так можно? Вот где проблема. Причем не могу сказать, что у меня есть решение.

Тезис третий – необходимость теории. Один мой бывший однокурсник – сейчас профессор Западно-Берлинского технического университета. Я бывал у него, и он у нас бывал несколько раз. Я однажды его спросил, чему учат у них в университете. Выяснилось, что изучают и логику, и все остальное, но только формулировки теорем. Я его спросил: «Скажи честно, если я сейчас подойду к како-



Поскольку никакого другого способа вычислить, кроме простого перебора, нет, — это трудноразрешимая задача...

му-нибудь вашему студенту и спрошу, что такое теорема Геделя о неполноте, он ответит?» «Нет, – говорит, – даже не вспомнит». «Тогда зачем так учить?» – «Ну, положено. А зачем вам теорема Геделя?» «Хотя бы для того, – говорю, – чтобы молодой специалист имел представление о границах применимости теории. Теорема Геделя говорит о том, что корректность арифметики не проверить средствами самой арифметики, и дает теоретические ограничения, предлагает искать какие-то метатеории, привлекать дополнительные возможности. Если человек об этом даже не подозревает, он будет в каких-то местах напрасно тратить время. У меня был случай, когда один выпускник кафедры математической физики, работающий у нас, должен был реализовать анализ потоков данных в программе. Он довольно быстро все реализовал. Самая мощная машина тогда была 486-я, и он на ней 4 часа тест из 20 строк гонял. Я посмотрел программу – простой перебор путей в графе. Я его спрашиваю: «Ты разве не знаешь, что число путей в графе растет экспоненциально относительно числа вершин?» «Не знаю. Подумаешь, экспонента! Машина железная, все посчитает». Я ему долго читал лекцию про актуальную бесконечность, о том, что если в программировании видишь экспоненту, то надо искать другое решение. Это не значит, что надо сдаваться. Я часто привожу студентам такой пример. На конференции, по-

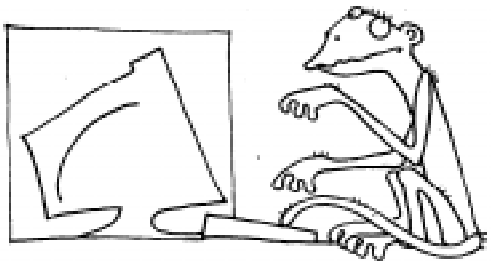


А вдруг найдется какой-то другой метод...

священной 1000-летию алгоритма, в Ургенче (на родине Аль-Хорезми), была представлена статья Ю.В. Матиясевича «Что нам делать с экспоненциально сложными задачами?» Это мне нравится, это конструктивный подход. Не просто «Все, сдаюсь, больше ничего сделать не можем». Всегда можно найти частные случаи. Есть и другая противоположность «теоретического» восприятия задачи. Другой не менее известный ученый меня мучил, когда я сдавал кандидатский минимум: что значит теорема Геделя о неполноте? И заставлял меня на экзамене (за две недели до защиты диссертации!) признать, что из этого следует, что машина не все может. Но это не так! Нет общего подхода – найдем частные.

Например, знаете, на чем основана шифрация? Сводят задачу к какой-нибудь трудноразрешимой (например, разложению числа на множители). Мои коллеги, практики, сделали систему шифрации. Популярная система, продается хорошо. Они меня попросили показать кому-нибудь из коллег – насколько их работа теоретически обоснована. Я попросил Ю.В. Матиясевича посмотреть их статьи. Он минут 10 смотрел, тут же указывает мне фразу: «Поскольку никакого другого способа вычислить, кроме простого перебора, нет, – это трудноразрешимая задача». Мне даже обидно стало, что сам не разглядел. А вдруг найдется какой-то другой метод, который для данного конкретного класса задач даст хороший алгоритм? Тогда все это рассыплется, как карточный домик. Возможно, такого метода и не найдется, но математика отличается тем, что все надо доказывать. Они не доказали, что другого метода, кроме прямого перебора, нет.

Итак, нужны теоретики, нужны исследования и нужны доказательства. И уж, если человек говорит, что эта система шифрации стойкая, – будь любезен, докажи, что никакого способа, кроме полного перебора, нет. Другое дело, что и в некоторых классических задачах криптографии этот вопрос до сих пор открыт.



...она не знает, что программировать...

Если у тебя нет теоретической подготовки, то так и будешь перебирать пути в графе, не задумываясь об экспоненциальной сложности алгоритма. На практике это означает, что алгоритм работать не будет. Теория дает определенные границы: за что браться, за что не браться, где искать, где не искать. Где с самого начала надо искать срез, подзадачу, специальный случай.

Все-таки это знание составляет малую часть нашей профессии, 5–10%. Просто есть вещи, которые надо знать. Если ты их вообще не знаешь, можешь налететь на такие грабли, что лоб себе расшибешь.

Давайте, все-таки, поговорим о программировании. Я много раз читал лекции в Гамбурге и в Классическом университете, и в Техническом университете. Там даже поговорить о программировании часто не с кем. Две крайности: или «коробочники», которые умеют пользоваться стандартными программами, или теоретики, которые занимаются чем-нибудь таким, что неизвестно, когда на практике осуществится. А людей, занимающихся нормальным программированием, часто и не встретить.

Я приведу еще один пример. Примерно в 1975 году мы получили первую ЕС ЭВМ 1030 среди гражданских организаций СССР, об этом даже в газетах писали. Первые ЕС ЭВМ шли только на оборону. И вот ленинградский математико-механический факультет получил самую первую машину за счет того, что мы делали много программ для ЕС ЭВМ. Машина часто ломалась, а мы сидели вечерами и даже ночами. Весь чай был выпит, все возможные темы обсуждены, И вот я начал одну девушку-оператора учить своему любимому языку АЛГОЛ-68. Такой сложный язык программирования, и редко какой студент мог его освоить в полном объеме. За несколько месяцев я научил ее так, что не каждый студент мог с ней сравниться. Говорю ей: «Теперь тебе надо переходить работать программистом. И зарплата выше, и работа интересная. Ведь что такое работа оператора? Поставить диск, загрузить машину». И тут я с ужасом понял, что она не знает, *что* программировать. Она не знает, как можно итеративно вычислить квадратный корень, она не знает, как устроен транслятор. Она знает язык программирования, экзамен сдать может, а программировать не может. На меня этот эпизод произвел очень сильное впечатление.

Прошло 25 лет, вроде бы многое изменилось. Но посмотрите, мы с Вами здесь сидим, каждые 10 минут дверь открывается. Каждый третий приходит с вопросом: «Андрей Николаевич, я хочу у Вас работать. Я слышал, что у Вас много людей занимается интересной работой». «Отлично, что ты умеешь?» «Я умею программировать на Паскале». «А что ты знаешь-то?». «Ну, как, я же научусь». «Посмотри на список спецкурсов кафедры. Что из этого ты знаешь?». «Ничего». «И как ты будешь работать? Я тебя определю в группу «Телефония». Ты знаешь, что как устроено? Что ты там будешь программировать? Ты умеешь писать a:=b, a:=b+c, но ведь это не программирование. Надо знать, что программировать».

Результатом таких разговоров может быть одно из двух. Кто-то всю жизнь меня после этого ненавидит за то, что он хотел заниматься интересным делом, а злобный Терехов на него ушат холодной воды вылил, а бывают такие упрямцы, которые говорят: «Ничего, я научусь». Хорошо, первые три месяца – стажировка, и, если выяснится, что человек работать не может, ему просто вежливо скажут: «Изви-

ни, друг, нам надо двигаться дальше». Это не значит, что человек пропадет, может быть, он в другую группу попадет. Бывает, что люди с третьей попытки свое место находят. Бывает так, что человек научится в процессе работы, но это скорее исключение, чем правило. Это обходится большими усилиями, чем у студента в процессе учебы, но зато закрепление совсем другое и мотивация другая.

Итак, между собой взаимосвязаны теория, практика, границы применимости теории, некоторые личные знания. Я объясняю своим студентам, что системный программист – это сфера обслуживания. Мы не делаем конечных продуктов. Например, человек производит расчеты. У него есть какой-то результат. При этом он пользуется трансляторами, операционными системами, вычислительными машинами, которые придумали другие люди. И системщики – как раз те люди, которые делают трансляторы, инструментальные средства, разрабатывают методологии по их использованию. А потом уже прикладные программисты этими средствами и методологиями пользуются и получают конечный результат. Очень часто, кстати, люди видят только конечный результат, особенно когда идет речь о делении денег, и совсем не видят той дороги, тех трудностей, которые были преодолены, чтобы этот результат получить. Системное программирование – это довольно старое название. По-русски мы хорошо знаем, что такое системное программирование. Но были проблемы, как перевести это словосочетание на английский язык. Есть наука computer science, она более теоретическая. А есть наука software engineering. Вот я заведующий кафедрой software engineering. Engineering – это разработка программного обеспечения. И мне кажется, что это довольно четко сейчас определилось.

Теперь следующий тезис, следующая проблема. Вот защищается мой аспирант у нас на Совете. Я сам член Ученого Совета. И каждый раз попадается какой-нибудь, мягко сказать, недоброжелатель, который, поскольку на Ученом Совете может выступать кто угодно, говорит: «Вы знаете, я не понимаю, почему это математика. Нет теорем, нет доказательств сходимости, почему эта диссертация защищается на математическом факультете?» В реальной действительности самый лучший ответ – сказать: «Вы ничего не понимаете в этой области». Времена, когда математика была только в теоремах, кончились как минимум 100 лет назад, а может быть, и больше. В середине XIX века Ч. Бэббидж придумал вычислительную машину, в которой были все основные элементы (процессор, память, программа, хранящаяся в памяти). Дочь Байрона Ада Августа леди Лавлейс пятистраничный доклад на итальянском языке этого Ч. Бэббиджа преобразовала в 100-страничный английский текст, где впервые ввела слова «Переадресация», «Процедура», «Цикл» – это что, не математика? Это было сделано более 150 лет назад.

Проблема решается просто, если говорить серьезно. Есть специальность – «Математическое обеспечение вычислительных машин, сетей и систем». Это не математический анализ, не математическая физика. Есть критерии, предъявляемые к кандидатской диссертации: новые результаты с практическим внедрением, языки программирования, их реализация, операционные системы и, самое главное, – модели. Математика начинается тогда, когда мы можем что-то формализовать, когда мы можем сформулировать задачу настолько точно, что можно построить алгоритм.

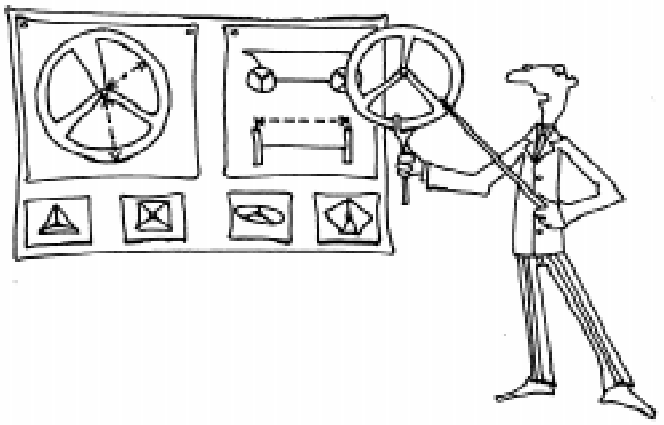
Почему возникли потребности в строгой формализации? Потому что пришло понимание, что некоторые задачи в принципе нельзя решить, потребовалась алгоритмическая формализация, и с помощью этих формальных методов удалось доказать неразрешимость нескольких проблем. Первые такие доказательства были корректно получены в 30-х годах XX века. Почему, когда я доказал про что-то, что

этого нет и быть не может, – это математика, а когда я построил формальную модель и по ней сделал алгоритм, который работает, – это уже не математика? Никто не сказал, что только отрицательные результаты являются математическими. В нашей стране такой консерватизм особенно силен. Граница между «наукой» и «не наукой» в данном случае, когда речь идет о программировании, достаточно понятная, но трудно формализуемая, поэтому вызывает массу проблем, профессионал их знает. И постоянно на Ученом Совете кто-нибудь начинает возражать, что это «не наука». Важно создать новую модель, новый язык, новый метод, новый алгоритм, показать, что он отличается от других. Я уже много раз был оппонентом, а не только руководителем диссертации. Для новичка, для человека со стороны это, может быть, будет даже удивительно. В задачу оппонента входит не только оценить, хорошая диссертация или плохая. Главная задача – оценить соотношение этой работы с другими известными работами. Не забыл ли диссертант, что это уже сделано? Он сравнивал свою работу с другими? Не забыл ли он какой-то важный результат, который был получен другим, а у него даже не упомянут? Насколько корректно проведено сравнение собственной работы с другими известными работами? Есть такая наука, которая называется software engineering, в ней есть своя область деятельности, свои требования, свои критерии. Профессионалы знают, что сделать хороший транслятор – это очень трудная задача. Сделать технологию, которой будет пользоваться широкий круг людей – это тоже очень трудно.

У меня защищалось в этом году 19 человек. Приходит студент V курса, проведший полгода где-то на практике. Показывает свою программу. Я говорю: «Хорошо, давай посмотрим, насколько научна твоя работа. Что здесь самое главное? Во-первых, насколько это новый результат? Писал ли кто-нибудь об этом раньше?». «Не знаю». «Ты делал обзор литературы?». «Нет»... Чем отличается исследователь от практика в худшем понимании этого слова? Исследователь поймет, что не надо изобретать велосипед. В наше время, когда есть Интернет, другие каналы получения информации, имеет смысл посмотреть, что сделали другие. С этого надо начинать. Далеко не всем это приходит в голову. Нашел, что такого результата нет, и ты его сам получаешь. Но есть похожие результаты. Проведи сравнительную характеристику. Мне не нравится такая ситуация: люди работали, писали дипломы, писали диссертации. Спрашиваешь: «Чем ваша технология лучше, чем остальные?» И следуют аргументы: «С одной стороны, нельзя не признать, с другой стороны, нельзя не согласиться...» Спрашиваю: «Ребята, вы потратили на эту работу столько сил и времени. Мы интуитивно понимаем, что это хорошо. Но разве так трудно все это четко сформулировать? У многих программ есть демо-версии, и сравнение их с вашим вариантом входит в работу». Нынешняя молодежь с трудом понимает, что просто сделать что-то, что работает, – это меньше половины дела. Настоящий исследователь должен смотреть, как работают другие, скачивать демоверсии, читать инструкции – как у них запускается программа, научиться запускать, посмотреть, поработать, понять, что хорошо, что плохо, может быть, заимствовать какие-то идеи. Необходимо четко сформулировать, в чем твоя заслуга. Что ты такого сделал, чего у других нет? Сейчас этим занимаются только бедные диссертанты, и то, к сожалению, есть такая десятилетиями сложившаяся практика, что они занимаются этим в последний момент, когда уже «кирпич» пишут. Положено иметь обзор литературы по теме, и они и сидят в библиотеке по 3 месяца в самом конце. Между прочим, на моих глазах была ситуация, когда аспирант защищается по отладке, а его на Совете спрашивают: «Как это соотносится с такими-то методами отладки?» Оказалось, что он в библиотеке Академии Наук сравнивал

свой метод с американскими работами, а то, что в СССР есть такие работы, и не знал...

Сейчас я попытаюсь еще раз сформулировать эту идею. Дали тебе работу – не поленись, потрать время, посмотри, что есть у других. Какие есть методы. Не изобретай велосипед. И в то же время – никакого низкопоклонства перед Западом, даже если там есть какие-то публикации. Посмотри, насколько они хороши. Если хороши – используй, со ссылкой. Для этого и существует вся мировая наука, чтобы пользоваться ее результатами. Не надо ничего высасывать из пальца. Но плохо, когда человек не потратил времени на то, чтобы лишний раз посмотреть, а потом ему кто-то другой указывает на точно такие же, если не лучшие, результаты других исследователей.



Он был уверен, что никто в мире этого не знает...

Итак, важно умение сформулировать задачу, умение посмотреть на нее со стороны «что есть в мире», умение четко объяснить, что у тебя нового.

Я в очередной раз отвлекся, но все-таки закончу. Приходит студент из мелкой фирмы с программой. Никакого анализа не сделано, сравнительных характеристик не сделано, никаких моделей не построено, ничему новому он не научился, то есть просто сидел и работал на зарплату. Что теперь делать? Я уверяю, что это не может быть дипломной работой. Дипломная работа обязательно должна нести элемент научного творчества.

Преддипломная практика – на первом семестре пятого курса. И когда человек нормально практику прошел, чему-то научился и пришел с результатом, у него еще есть время на то, чтобы превратить эту работу в дипломную. Бывают смешные случаи. Два студента сделали транслятор с Java. Никаких ссылок – будто бы в безвоздушном пространстве. «Сколько вы знаете трансляторов с Java?» «Один или два». Поискали в Интернете – оказалось, 20–30. «Дайте сравнительные характеристики». Выяснилось, что проигрывают в 500 раз одному из трансляторов. Говорю: «Ребята, я читал про унтер-офицерскую вдову, которая сама себя высекла, но не думал, что это может быть среди студентов мат-меха». Посмотрели программу – это оптимизирующий транслятор с языка C. Java – это интерпретатор, по определению, поэтому их и сравнивать по эффективности нельзя, это задачи с совершенно разными целями. Умение грамотно сформулировать задачу здесь было необходимо. Хорошо, что я успел увидеть эту дипломную работу, иначе над нами бы еще долго смеялись.

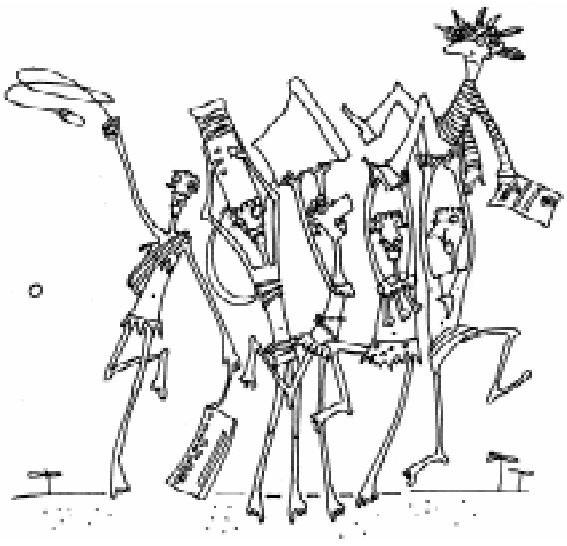
Или другой пример. Приходит студент с работой по теме «распараллеливание». Я не был научным руководителем, но, прочитав работу, почувствовал, что я такое сто раз слышал. У меня в это время дочь в Англии была и как раз занималась распараллеливанием. Позвонил ей, она перечислила методы и средства, которые сейчас используются. На мальчика было жалко смотреть. Конечно, он получил нормальный результат, что-то сосчитал и думал, что изобрел новый метод. Он был уверен, что никто в мире этого не знает. Откуда такая уверенность? Конечно, мы учим своих студентов тому, что «мат-мех лучше всех», но всему есть свой предел. Так же нельзя, ты же не один в мире.

Итак, следующий тезис такой. Мы готовим университетских людей, которые должны быть способны – я на этом жест-

ко настаиваю – не только написать программу и придумать эффективный алгоритм, но и сравнить, обосновать, понять теоретически возможные границы применимости. Это обязательно входит в университетское образование.

Но кто сказал, что только такие люди нужны? Мы переходим к следующему вопросу. Вот передо мной лежит проект программы «Федеральная целевая программа электронной России на 2002 – 2010 годы». Мне очень понравилась фраза: «В настоящее время наблюдается определенное перепроизводство «массовых» специалистов с высшим образованием, в частности, в области создания программного обеспечения и технической поддержки информационно-вычислительных и коммуникационных систем, которые часто вынуждены выполнять функции, не требующие широкой теоретической подготовки, которые могут и должны замещаться специалистами со средним образованием». Еще год назад, в связи с конференцией по Наукограду, я изучал материалы про аналогичные зоны в Индии, в Ирландии. В Индии готовится в год 5000 master of science (аналог нашего высшего образования) и 27000 бакалавров. Мы пытались прикинуть, сколько у нас готовится. Получилось несколько тысяч программистов с высшим образованием очень разного профиля. У нас и железнодорожный институт, и институт водного транспорта готовит программистов, но все равно получается всего несколько тысяч. А бакалавров нет вообще. У меня, как видите, вся стена увешана досками с именами наших сотрудников, которые готовят команды на международные олимпиады по программированию. Пять лет подряд в десятке из 2500 команд, два года подряд чемпионы. Это приятно. Но только за 5 лет ни один из них не стал профессиональным программистом. Они математики. Сейчас Андрей Лопатин перешел на нашу кафедру и переучивается. Нельзя сказать, что у него все так бравурно идет, он тратит силы, время, и надеюсь, что из него мы сделаем профессионала. Это первый такой пример за 5 лет. Я и раньше пытался сказать, что мы слишком сильно сконцентрировались на результатах олимпиад. Действительно, важно побеждать на олимпиадах, важно иметь исследователей. У меня на предприятии есть несколько человек, таких, что, если хотя бы один из них уйдет, мое предприятие лопнет. Есть люди, на которых держится предприятие, потому что они придумывают новые идеи, они все цементируют. У меня на предприятии больше 200 человек, а таких, без которых все остановится, 2–3, может быть, 5, если молодых прибавить. Если уйдут другие – можно найти им замену. Но именно остальные 200 человек выполняют основной объем работы. Бакалавриат у нас как-то не прижился, но нужно найти какие-то упрощенные формы подготовки программистов.

Иногда говорят: «У нас такой отсев, у нас уезжают на Запад». У нас на кафедре немногие уезжают – за все годы человек 15 из 250–300 выпускников уехали за рубеж. Многие из этих 15 не просто так уехали, а по согласованию со мной перешли в компании в США, которые со мной же сотрудничают. То есть можно сказать, что эти выпускники со мной же и работают, только с другой стороны. Кста-



У нас такой отсев, у нас уезжают на Запад...

ти, это очень помогает. Всегда приятно, когда там свои люди, говорящие по-русски, с нашим менталитетом. Дело не в английском языке, у нас по-английски все говорят, но, когда там сидит человек и может поговорить на «ты» с кем-то из наших, это всегда сильно помогает. Так что это даже нельзя назвать потерей. Бывает отсев, когда уходят в другие компании. Бывает так, что выпускник нашей кафедры уходит за зарплатой в 1000 долларов. Я столько платить не могу. А они уходят и часто страдают. Вот сейчас Artificial Life объявила, что закрывается, другие компании сокращают количество сотрудников. Я не то, что мстительный, я слежу за своими учениками. Этот был вынужден уйти туда, этот туда. Не все у них плохо, слава Богу, но за длинным рублем гнаться не стоит, особенно в молодые годы.

Еще немного отвлекусь, уж больно пример для меня интересный. Когда уезжали хоккеисты уровня Ларионова и Фетисова, это было одно. Люди, сделавшие имя, они за рубежом хорошо известны. Когда уезжают 19–20-летние ребята, чтобы годами играть в фарм-клубе, то потом, даже когда пытаются вернуться, уже здесь не могут играть на хорошем уровне, потому что там они не играли у таких великих тренеров, как Тихонов, Тарасов, и имели мало игровой практики. Аналогия абсолютная. Одно дело – человек закончил математико-механический факультет, еще лучше – аспирантуру, защитился, получил 5–10 лет стажа, поработал, принес пользу своему предприятию, своему родному университету, научил молодых специалистов и поехал. Вперед, я даже помогу. Это профессиональный рост, это интересная работа. Я вовсе не к тому, что нельзя ездить. Он там получит новые знания, новые силы, новую информацию для себя и, возможно, для нас. Может нам принести заказы, интересные научные исследования. Все это понятно. Я считаю, что это нормально. Совсем другое дело, когда уезжает совсем молодой человек. У меня был случай, когда студент полгода недоучился, диплом не защитил и уехал. Ведь диплом Ленинградского – Петербургского университета во всем мире признается. «Вы извините, может быть, мне перейти на другую кафедру, чтобы вашу кафедру не позорить?». «Да, – говорю, – и так уезжай!» Но я считаю, что это уже перебор. Кому он там нужен? Сейчас там перепроизводство программистов, массовые сокращения. Опять-таки не подумайте, что я злорадствую. Хотя, конечно, тех, кто меня бросил, тем более «на полуслове», не передав материалы, не передав информацию, я тоже запоминаю. Были разные неприятные случаи, но мне не хотелось бы о них подробно рассказывать.

Должны быть какие-то законы. Я в начале разговора вспоминал про федеральную составляющую. Я не могу этот закон обойти. Если я его не выполню, то диплом будет считаться недействительным. Эти вопросы надо решать на государственном уровне. Ровно такая же ситуация и с отъездом молодых специалистов. И она, кстати, решается. Не решена, но решается. Если этим интересуетесь, поищите в Интернете. Во многих странах дается кредит на образование, который ты должен отработать. Если уезжаешь – будь любезен, отдай. Неужели мы настолько богаты, чтобы об этом не думать? Должны быть определенные жесткие государственные законы. Если студент или его родители платили по 5000 долларов за семестр – уезжай, куда хочешь. Но, если на тебя государство истратило столько денег, – почему мы должны отпускать человека, который у нас получил образование?

Я понимаю, что у человека, занимающегося классической математикой, могут быть проблемы. Один мой приятель, алгебраист, который недавно уехал, говорил: «Если бы наше государство хотя бы треть тех денег, что в США, платило, я работал бы здесь. Пойми меня, я не хочу заниматься программированием». Все понятно, я тоже не хотел бы заниматься ал-

геброй. А это фундаментальная наука, и государство ее практически не поддерживает. А у него семья, двое детей, вот он и уехал. Я в него камнем не кину. У него выхода не было, его государство так подставило. И не он один. Но это «классическая наука». А люди, которые занимаются прикладными вещами, живут. Причем, не только программисты. Я сейчас общаюсь, например, с людьми, которые корабли строят. Есть заказы, есть интересная работа, есть зарплата.

Если программист бедный, – значит, он плохо работает. У нас такая специальность, что на кусок хлеба заработать можно. Для меня отъезд за границу не представляется серьезной проблемой. Когда говорят про «утечку мозгов», это скорее лозунги для демонстрации. Создайте рабочие места, обеспечьте интересной работой, обеспечьте приличную зарплату (пусть не на уровне США, но так, чтобы здесь можно было содержать семью), и почти никто не уедет. Поедут только те, кто любит приключения.

Что такое государственная поддержка? В Ирландии, если предприятие делает программное обеспечение, оно платит 10% налога с прибыли при норме 28%. Но это относится только к прибыли, полученной от программного обеспечения. Программное обеспечение отличается в лучшую сторону от многих других видов продукции тем, что это возобновляемый ресурс. Индия, благодаря развитию телекоммуникаций и программного обеспечения, получает большие деньги с экспорта, в страну возвращаются те, кто ранее ее покинул. Ирландия – маленькая страна, но производит 40% программного обеспечения в Европе. Десять лет назад это было трудно предположить, но многое решает разумная государственная поддержка. Все должно быть в разумных пределах.

Государственная поддержка – важный момент, и есть положительные примеры – Индия, Ирландия, Израиль. Имеется в виду не только налоговая политика. Например, хорошо подобранная программа обучения – это государственная поддержка. Инфраструктура телекоммуникаций – это государственная поддержка. Удобный проезд до крупнейших учебных заведений – тоже государственная поддержка. Вспомните наш мат-мех! У нас мог бы быть совсем иной контингент студентов, если бы электричка ходила полчаса, а не час, как сейчас. В Сингапуре в Школе программирования на IV курсе занятия начинаются в 17 часов. Дело в том, что, если ты не работал на фирму полный рабочий день, – диплом не выдадут. Какой ты программист, если никогда не работал профессионально? И это на дневном отделении. Сначала поработай, и пусть отзыв дадут, что ты действительно программист.

На юге Швеции в некоторых маленьких городках, по 30 тысяч жителей, есть свои IT университеты. Например, пять факультетов: программирование, технологии, архитектура телекоммуникаций, экономика, юриспруденция. Отделения в Англии, в Техасе, много профессоров, и программы хорошие, я сам их смотрел. В Швеции каждый университет окружен технопарком, 200–300 мелких компаний. Государство предоставляет здание (побольше нашего мат-меха). Аудитории, места для конференций, дешевые гостиницы. Получается такой симбиоз: университет готовит специалистов, которые работают в этих фирмах. Технопарку выгодно иметь ресурсы, а университету выгодно иметь финансовую поддержку и обратную связь, чему учить и как учить. Я уже давно говорю, что в нашем университетском городке тоже надо бы сделать технопарк. Пока никак не получается.

И еще одна тема – развитие собственно науки. Есть класс людей, которые готовы работать за меньшие деньги, но при условии, что будут заниматься наукой. Это особенность психики человека, которому неинтересно делать программное обеспечение с заранее указанными свойствами в жесткие сроки. Но таких людей мало. И проблема в том, что

в Петербурге есть сотни компаний, где платят приличные деньги, и программист может просто зарабатывать, но предприятия наукой заниматься не хотят.

Мы не можем все свести к работам «по заказу», даже если работа хорошо организована, поддержана мощными современными технологиями и налажены все организационные процессы. Кто будет создавать новые идеи, новые методологии, новые инструментальные средства? На мой взгляд, есть определенный процент выпускников университета, у которых в особенностях психики заложено, что они хотят сидеть в тиши лаборатории и заниматься наукой. Это люди, которые готовы получать, скажем, в два раза меньшую зарплату, чем люди на фирмах, но заниматься своей любимой наукой.

Кстати, первый раз я столкнулся с этой проблемой в Гамбурге. Это было в 1989 году. Хорошие преподаватели Гамбургского университета получают 7000 марок. Я навел справки и выяснил, что можно работать в программистской фирме и получать 20000 марок тут же, в Гамбурге. И места есть, и приглашают. Но преподаватели на такую работу не стремятся попасть. Я постарался узнать, почему. Ответ был следующим: «Во-первых, 7000 марок вполне достаточно, у меня есть дом, у меня есть дача, машина, что мне еще надо? В бочку, что ли, эти деньги складывать? А во-вторых, в программистской фирме будет жесткое управление, жесткие приказы. Я этого не люблю. Я работаю в университете. Этой зарплаты мне достаточно, чтобы прилично существовать, но я хочу заниматься именно наукой».

Я понимаю этого преподавателя. Есть такие люди. Если бы не было таких людей, то наша наука бы остановилась. Где им работать здесь, в Санкт-Петербурге? Какая фирма возьмет на себя обязанность материально поддерживать фундаментальные исследования?

Кстати, а что это такое – фундаментальные исследования? У моих прагматичных друзей-американцев есть забавный, но полезный критерий, как отличать фундаментальные исследования от нефундаментальных. Если исследование 3 года не приносит прибыль, оно считается фундаментальным, и та часть прибыли, которая в него вложена, освобождается от налогов. И фирме это полезно. Если же, не дай Бог, через 2,5 года из этого исследования получился результат, который был применен на практике, продан, получены деньги за него – налоговый инспектор пересчитает налоги за него. Значит, исследование не было фундаментальным. Можно улыбаться, глядя на это правило, но оно существует, и оно работает. В США очень много забавных правил, но еще более забавно, что основная масса людей этим правилам следует.

Так, или похожим образом, или совсем по-другому, но можно придумать законы для поддержки фундаментальных исследований. Например, я директор предприятия. У меня есть свой директорский фонд, то есть сравнительно небольшая сумма денег, которую я отнимаю от других заказов и имею право тратить их по своему усмотрению. Практически все эти деньги я трачу на поддержку исследований.

Но те люди, которые получают деньги, чувствуют себя «людьми второго сорта», потому что время от времени я вынужден им напоминать, что мне нужны результаты, что я ради вас брал деньги у такой-то группы... Это не улучшает психологический климат. И те люди, у которых я отнял деньги, тоже не рады: «Лучше бы ты нам еще два компьютера купил, лучше бы ты нам новую мебель купил» и так далее. Всегда есть необходимость что-то купить, разумно истратив деньги для той группы, которая их заработала. У меня даже распадались группы, и одной из причин ухода сотрудников было их несогласие с моей политикой распределения денег. Они твердо настаивали на том, что они заработали деньги, поэтому эти деньги должны быть истрачены на их зарплату, в крайнем случае, на их инфраструк-

туру, на компьютеры. Даже, может быть, и на исследования, но только в этой узкой области, где они заработали деньги. И все мои попытки объяснить им, что так не бывает, что никто не может предсказать, какое направление в науке проявит себя через два-три года, ни к чему не приводят. Я уж и не говорю о том, что понятие «они заработали» – совершенно неправильное. Зарабатывает предприятие, директор, маркетологи, завхоз, сетевые администраторы и т.д. Когда «уговаривают» заказчика, ссылаются на предыдущие работы предприятия. Заказчик с большим удовольствием работает с устоявшимся большим коллективом, а не с малой группой, которая в любой момент может разбежаться.

Я не настаиваю на том, что мое решение единственно правильное. Возможно, в чем-то и они были правы. Но, тем не менее, я искренне считаю, что большое предприятие, диверсифицированное, то есть имеющее много направлений работы и много направлений исследований, стабильнее и надежнее. Бывает, что некоторые направления, даже очень многообещающие, неожиданно коллапсируют. Нынешние резкие падения коэффициента высокотехнологичных производств в Америке, по-моему, как раз это и показали. Сколько было ожиданий от всех этих бесконечных интернет-приложений! А теперь идет массовое закрытие фирм, увольнение людей. Может быть, найдется 2–3 умных человека, которые скажут: «Я об этом и говорил», только я таких не знаю. Трудно предугадать, что разовьется, а что нет.

Я считаю, что мы (тем более мы, выпускники мат-меха!) должны работать в возможно более широком наборе направлений. Во-первых, это интересно, во-вторых, таковы законы науки, а в-третьих, это обеспечит определенную стабильность, если что-то все-таки по не зависящим от нас причинам или из-за нашей плохой работы не оправдает надежд. В этом случае маленький коллектив обанкротится и будет вынужден искать другие способы существования, а большие предприятия это спокойно переживут.

В конце концов, я пришел к пониманию, что поддержка науки не может быть выполнена на уровне спонсорской поддержки, на уровне отдельных пожертвований. Как и везде, нужна системность, нужна структура, нужны определенные «правила игры».

Сейчас я прилагаю много усилий для создания института, который так и будет называться: научно-исследовательский институт информационных технологий, входящий в структуру моего родного Санкт-Петербургского государственного университета. Я провел беседы со многими исследователями нашего факультета, в частности, и с такими, которые весьма далеки от практики, но продуктивно работали в нашей области и в computer science. Мы с ними наметили несколько направлений, в которых исследования особенно важны сейчас, даже без стопроцентной уверенности, что через два-три года они приведут к видимому результату, который можно будет использовать. Я получил поддержку Ученого Совета факультета, хотя тоже не без вопросов. Вопросы были такие же, как несколько лет назад, когда создавалась новая кафедра: «Зачем нужно еще что-то новое? Нельзя ли справиться старыми структурами?» Но, тем не менее, процесс идет.

Я надеюсь, что мне удастся этот институт создать. Более того, под этот пока не созданный институт я нашел западную компанию, которая готова материально поддержать проект. Причем я их не обманывал. Я говорил, что институт будет заниматься исследованиями и не обещал немедленной прибыли. Тем не менее, крупные компании понимают, что если поддержат исследования, то все равно получат выгоду. Если появится положительный результат, он, в первую очередь, будет применен в их интересах. Это нормальный мировой процесс хоть в Западной Европе, хоть в США. Любой ученый из любого университета до 40% своего вре-

мени тратит на поиск грантов, их обоснование. Не считается зазорным делать 3, 5, 10 попыток. В одном, в другом, в третьем фонде.

В первую очередь, нужна структура для такого поиска. Нужны специальные люди, которые следят, где объявлены гранты, где объявлены программы, какие к ним предъявляются требования. Именно эти люди должны оформлять бумаги. Сначала надо сообщить: «Коллеги! Есть такие-то предложения. Требования такие-то, сроки такие-то». Специальные люди должны оформлять и рассылать бумаги (ученые не всегда удачно их оформляют). Должна вестись база данных: на что получен положительный ответ, какие замечания были, как на них реагировать. Этим должны заниматься специально подготовленные люди, а не сами исследователи. И я думаю, что в наше время научно-исследовательский институт – это как раз та структура, которая сможет сконцентрировать внутри себя исследователей, обеспечить им определенную поддержку для того, чтобы получить финансирование, отслеживать сроки и этапы исследований.

Короче говоря, все должно быть сделано для того, чтобы ученые максимум своего времени тратили на исследования, получали за это достойную зарплату, чтобы не надо было «халтурить», отвлекаться на различные работы. Мешки с картошкой и песком сейчас никто уже не грузит, но случаи, когда заслуженные ученые, профессора вынуждены заниматься неинтересной работой, далекой от науки, только для того, чтобы получить зарплату, я знаю. Их много.

Кто знает, каким получится этот институт? Но я искренне верю, что институт с международным участием поможет тем ученым, которые предрасположены к научной деятельности, не только выжить, но и активно участвовать в развитии нашей любимой науки.

Терехов Андрей Николаевич,
профессор, заведующий кафедрой системного программирования математическо-механического факультета СПбГУ