Редкая *профессия:* русские «плюсы»

Евгений Зуев

(Продолжение, начало см. в PC Magazine/RE, спецвыпуск 5/1997)

...Наступило «междуцарствие». Так можно сказать уже сейчас, по прошествии многих лет, когда уже известно, «чем дело кончилось», — тогда же ситуация воспринималась как тупиковая. Что делать дальше — было совершенно непонятно. Вроде бы все закончилось, проект завершен — стало быть, оставить компилятор и заняться чем-то другим?..

Это казалось невозможным. Как сама компиляторная тематика, так и конкретный проект «не отпускали», и дело не только в приверженности этой тематике и привязанности к нашему компилятору. Я уже писал о мощной инерции написанного программного текста, подобной силе, которая тянет вперед тормозящий поезд. Это ощущение мы в полной мере испытали на себе, когда, казалось, уже ничто не заставляло нас упираться. Мы продолжали и продолжали работать над ним. Каждый день что-то правили, улучшали...

От редакции

В журнале PC Magazine/RE (спецвыпуск 5/1997 г.) была опубликована статья Евгения Зуева «Редкая профессия». В ней рассказывалось об истории создания ISO-совместимого компилятора Си++ по заказу европейской компании и о перипетиях, связанных с работой над этим проектом. Статья вызвала большой интерес, и даже сейчас — более чем через пятнадцать лет после появления — у нее есть читатели, которые исправно заходят на страницу с этой статьей в электронном архиве журнала *www.pcmag.ru/zouev*.

С тех пор в профессиональной жизни автора произошло много событий. Он смог поучаствовать во многих софтверных проектах, связанных с проектированием и реализацией различных языков и компиляторов. В издательстве «ДМК-Пресс» вышла его книга с тем же названием («Редкая профессия»), в которую вошли рассказы об истории некоторых из этих проектов.

Автор любезно согласился предоставить для публикации фрагменты своей книги, в которых обобщается опыт разработки компилятора Си++, а также описываются события, произошедшие с компилятором и с автором после выхода той давней статьи. Нам представляется, что современное продолжение этой истории ничуть не менее интересно и увлекательно, нежели ее начало.

Проект Wednesday

Однажды к нам в комнату пожаловала «делегация» из трех человек. Саша Кротов пригласил своих знакомых из московской фирмы, которая сделала среду программирования на Си для однокристальных микропроцессоров[1]: они попросили содействия в тестировании их Си-компилятора. Наш тестовый набор для «плюсов» включал так называемую «Си-часть», и ее-то ребята и хотели использовать для проверки своего компилятора. Мы не решились отдавать «на сторону» тестовый набор, который формально принадлежал не нам, и предложили проверить их компилятор на наших компьютерах.

Не помню, честно говоря, насколько успешными были их тестовые прогоны, но это оказалось неважно: мы быстро поняли, что у нас очень много общих профессиональных пристрастий, и общаться с ними было очень интересно и полезно. Помимо общей для нас всех «компиляторной» тематики, они обладали ценным и неведомым для нас опытом создания целостной среды программирования. Кроме того, ребята выгодно отличались своим глубоким знакомством с миром спецпроцессоров (а это весьма специфическая и очень широкая на самом деле сфера! — различных моделей «спецпроцессоров» в мире выпускается на пару порядков больше, нежели «обычных» микропроцессоров для персоналок, смартфонов или планшетов, и устроены они, как правило, иначе).

Постепенно наше общение приобрело регулярный характер: если сначала они приезжали к нам в МГУ, чтобы поработать с тестовым набором, то вскоре наши встречи оказались полностью посвящены обсуждению интересных для всех нас компиляторных и околокомпиляторных тем. Мы пили чай, курили, разговаривали, спорили и между делом рисовали на бумаге квадратики и кружки с надписями и соединяли их линиями и стрелками.

Мы рассказывали им, как устроен наш компилятор и как мы собираемся его модернизировать (и уже модернизируем). Они показывали нам свою среду и объясняли особенности устройства спецпроцессоров. Наша идея о промежуточном представлении (я настаивал на термине «семантическое представление», СП) как основе и центральной части любой языково-ориентированной системы много раз обсуждалась и в итоге получила общее одобрение. Они, в свою очередь, выдвинули идею интерпретации СП как альтернативного метода выполнения программ на Си++, который потенциально сулит существенные выгоды и преимущества.

[1] Именно эта фирма упоминалась в конце статьи «Редкая профессия».

А в качестве реализации этой идеи предложили сделать *виртуальную машину* Си++ — программу, которая бы обходила наше дерево программы и непосредственно исполняла конструкции языка. Мы думали, как должна выглядеть «настоящая», «правильная» среда программирования на языке Си++, которая бы превосходила по тем или иным возможностям аналогичные (западные) системы.

Со временем смутные сначала идеи приобретали более конкретные очертания, разговоры «обо всем» превратились в предметные и систематические обсуждения. Мы начали протоколировать ход обсуждений, чтобы не забыть со временем ничего ценного из наших «мозговых штурмов». Вместо рисования схем мы стали писать планы, придумали название для нашего гипотетического проекта — Wednesday. По той простой причине, что собирались мы обычно по средам.

Конечно, мы отлично понимали, что никаких перспектив у нашего проекта не было. Какие-либо конкретные шаги в этом направлении даже не обсуждались; никаких «венчурных фондов», куда можно было бы обратиться за поддержкой, тогда в России и в помине не было. На самом деле, нам всем просто хотелось «отвести душу» и, забыв на время о реалиях жизни, придумать, как такую «правильную» систему *можно было бы* сделать.

Мы не могли даже представить себе, что до момента превращения наших мечтаний в реальность оставалось совсем немного времени...

На следующий день

Статья «Редкая профессия» написалась, честно говоря, от безысходности: ситуация с компилятором была совершенно неопределенной. Понимание того, в каком направлении его развивать, и у меня, и у Саши Кротова отчетливо присутствовало, однако куда «пристроить» наш продукт и что вообще делать дальше, было решительно непонятно.

Кое-какие шаги мы пытались предпринимать: докладывали о нашей работе на паре конференций, а также с помощью и при посредничестве нашего шефа, профессора Владимира Александровича Сухомлина, встречались с представителями одной серьезной и уважаемой московской фирмы, рассказывали им о нашем проекте... Альянса не получилось: хотя компилятор Си++ был им реально нужен, в итоге они предпочли купить (или уже купили к моменту нашей встречи, не помню) аналогичный западный продукт. Конечно, развивать компилятор можно было и в рамках нашей работы в МГУ (мы, собственно, это и делали), однако его перспективы от этого не становились яснее.

В то время я немного подрабатывал переводами статей в PC Magazine/RE и довольно часто бывал в редакции — и как-то получилось, что в разговоре с главным редактором возникла мысль рассказать об истории создания компилятора Си++.

Интересно: когда я предложил журналу идею статьи (а к тому моменту текст был уже написан), меня предупредили: пиши, но не больше 40 килобайт (тематика журнала сказывалась: вместо привычных «авторских листов» или «тысяч знаков с пробелами» там оперировали «килобайтами»). Придя домой, я увидел, что мой практически готовый текст больше 100 килобайт. Бросившись его сокращать, я впал в ужас: было абсолютно непонятно, что выкидывать... В итоге махнул рукой и отдал 90 килобайт — не мог резать по живому, пусть сами сокращают. Напечатали целиком.

Хотя, как я сказал, при написании мной двигали не самые радостные чувства, я все-таки не связывал с выходом этой статьи каких-то особенных надежд. Поэтому то, что произошло потом, оказалось для меня совершенно неожиданным...

На *следующий же день* после выхода журнала со статьей случились два телефонных звонка. Звонили ребята из новосибирской фирмы XDS, которые разрабатывали компиляторы для Модулы-2 (с очень неплохими, кстати, производственными характеристиками). Никаких особенных поводов у них не было: они просто очень удивились, узнав, что еще кто-то в России, кроме них, сейчас делает компиляторы, и хотели познакомиться. Через некоторое время мы встретились и интересно пообщались.

А вот второй звонок имел куда более серьезные и продолжительные последствия. Говоря коротко, мне было предложено создать фирму, которая бы продолжила разработку компилятора с тем, чтобы сделать из него коммерческий продукт. Выражалась готовность вложить в разработку довольно значительную по моим представлениям сумму.

Во встречах, которые последовали после этого второго звонка, детали предложения были описаны более подробно. Во-первых, в качестве цели проекта было обозначено нечто вроде интегрированной системы разработки — с визуальным редактором, отладчиком и т. д. Что-то похожее на изделия Borland или Microsoft. Во-вторых, результат разработки позиционировался как *коробочный продукт* (!). И наконец, предлагалась радикальная локализация компилятора. Чтобы все было по-русски: диагностические сообщения, идентификаторы в программах и... служебные слова языка!

Ну и под этот проект следовало основать нормальную, «правильную» фирму; автор предложения хотел опробовать на этом проекте какой-то новый вид организации, отличный от других его компаний. Этот аспект интересовал меня в самую последнюю очередь, и я, честно говоря, не особо вникал в эти его идеи: тогда я не мыслил для себя никакой иной стези, кроме как разработчика. (Это равнодушие к бизнес-аспектам инициативы и было, наверное, моей самой большой ошибкой... Впрочем, об этом ниже.)

Первой реакцией был скепсис и несогласие со всеми содержательными предложениями. Конкурировать с западными инструментами разработки на их поле совершенно бессмысленно и заранее обречено на неудачу. Пусть наш продукт будет понятен российским пользователям — нам все равно не удастся за приемлемое время создать что-то сравнимое по качеству. Их системы создаются громадными коллективами (в команде, делающей компилятор Borland,— где-то видел

их групповую фотографию — около сорока человек!), с несравнимыми бюджетами и весьма долгое время. Создание российского коробочного продукта требует очень серьезных усилий, и вряд ли он будет иметь коммерческий успех: на следующий день после выхода он появится в ларьках у пиратов, и никому и в голову не придет идти за ним в магазин. (Западные фирмы-производители хотя бы могут надеяться получить доход у себя же на Западе, где пиратство не столь распространено. А что делать нашим разработчикам, да еще с продуктом, который принципиально ориентирован на российский рынок?..) Ну, а русификация, если и имеет смысл, то лишь в некоторых аспектах: диагностики, оперативная помощь... Перевод же служебных слов вызовет тотальную критику и издевательские комментарии (так впоследствии, в общем, и случилось) и не добавит привлекательности продукту. К тому же такое насилие над языком явно противоречит стандарту, к которому за последние годы мы привыкли относиться очень уважительно.

разработки, созданные на основе нашего компилятора Си++ для нескольких моделей отечественных микропроцессоров.

Автор предложения, казалось, был искренне расстроен моей реакцией. Надо пояснить, что его фирма (довольно известная и успешная в то время) занималась разработкой лингвистических систем: словарей, переводчиков и других подобных продуктов. Его логика как бизнесмена была примерно такова: сейчас мы делаем продукты для естественных языков, а компилятор — это программа, имеющая дело с искусственным языком. И там и там — языки. Если англо-русский словарь — это двуязычная структура, то почему компилятору не быть двуязычным? И коль в обоих случаях речь идет о языках, стало быть, для продвижения компилятора применима, в общем, такая же бизнес-модель, что и для словаря...

Как я уже сказал, высказанные идеи в общем не пришлись мне по душе, а кое-что показалось даже нелепым. Конечно, я не мог не выказать свое критическое отношение (впрочем,

По понятиям сегодняшнего времени, прдлагаемое предприятие было классическим стартапом.

С моей точки зрения, ближайшие коммерческие перспективы нашего компилятора лежали не в сфере коробочных продуктов для массового пользователя, а скорее в области разработки софта для всякого рода специальных процессоров. Имея одну и ту же «фронтальную» часть компилятора, достаточно написать генератор кода, который бы порождал выполняемый код для нового процессора, и получить тем самым законченный инструмент разработки. Забегая вперед, скажу, что в конце концов перспективность такой, вполне очевидной, на мой взгляд, стратегии в итоге стала ясной и другим участникам описываемых событий. Сейчас компания «Интерстрон» *(www.interstron.ru)* — непосредственный наследник наших проектов и инициатив — успешно реализует инструменты

весьма сдержанно). Но на самом деле мой скепсис не имел никакого значения и смысла: в глубине души я *уже согласился — сразу и на все*. Детали можно было обсуждать, но, продолжая слушать и отвечать, в мыслях я уже составлял план работы, прикидывал, как распределить обязанности, и даже думал, кого нам с Сашей привлечь к участию в проекте. И конечно же, первые, о ком я подумал,— наши коллеги по «виртуальному» проекту Wednesday.

Странно? Только на первый взгляд и для того, кто смотрит на это предложение со стороны и с позиций сегодняшнего дня (напомню, все это происходило в начале 1998 г.). Впрочем, я готов принять любую критику своих тогдашних неразумных действий: основная доля вины в том, что слишком многое пошло не так, как следовало бы, лежит на мне. Ну а чтобы избежать этой критики, давайте я сам попытаюсь рассказать обо всех своих ошибках и просчетах. Не подумайте чего-то ужасного: сразу предупреждаю, что ничего драматического не случилось, никто нас не обманул и не «кинул»[2]. И мы не обманули. Речь идет именно об ошибках.

По понятиям сегодняшнего времени предлагаемое предприятие было классическим стартапом. Была команда из двух неплохих профессионалов (и энтузиастов), которая сделала... ну, пока не продукт, конечно, но, скажем, очень хороший прототип продукта. Кстати, аналогов этому «прототипу» в России не было[3], при том, что потребность в отечественном компиляторе для самого мощного и распространенного языка объективно существовала. И вот нашелся инвестор (или это называется сейчас «бизнес-ангел»? — не знаю; не разбираюсь в тонкостях терминологии), который выразил готовность поддержать проект, помочь превратить его в коммерческий продукт, продажи которого приносили бы прибыль. Конечно, за средства, инвестируемые в создаваемую фирму, он претендовал на определенную долю акций компании.

Сегодня Интернет полнится текстами, детально описывающими шаги по поиску инвесторов для стартапов. Советуют, где их искать, как составлять презентации и бизнес-планы, в какие фонды следует обращаться (а каких избегать) и как общаться с потенциальными инвесторами. Уйма фондов предлагает содействие инициативным разработчикам. Полно конкретных историй удачных и неудачных стартапных инициатив. Читай, набирайся опыта, примеряй к себе и действуй!.. А тогда-то ведь не было ничего подобного, да и слова такого —

[2] Не забывайте, что эта публикация — не боевик (хотя на том же материале можно было придумать захватывающий авантюрный роман, лишь чуть-чуть «подправив» факты), а простой рассказ о реальных событиях.

[3] Замечу, что аналогов нет и до сих пор: наш компилятор как был первой в России полной реализацией Си++, так и остается пока единственной.

«стартап» — никто не слышал. И уж конечно никаких советов нам никто дать не мог. Не говоря уже о том, что мы никогда и не ставили себе такую задачу — «найти инвестора» — и не предпринимали для этого никаких шагов. Я просто взял и рассказал в статье о нашем проекте, ни о каких последствиях не думая... И вот, пожалуйста, немедленно и на блюдечке! Казалось бы, все отлично, и даже более чем.

Но правильно говорится: «дьявол в деталях».

Дьявол и детали

Предлагалась следующая конфигурация. Во-первых, создается новая фирма, акционерное общество, доля инвестора в котором 80%.

По сегодняшним меркам это выглядит как нонсенс; при создании стартапов, судя по публикациям, инвестор обычно претендует не более чем на 10–20%, и многие прямо пишут, что следует немедленно «посылать» подальше тех, кто хочет больше пятидесяти... Мне эта цифра была названа при первой же встрече — твердо и определенно. А в качестве обоснования было прямо сказано, что «мне нужен полный контроль».

Вообще-то дальше можно не писать. Все остальные условия на самом деле не столь важны, как это; именно здесь мы совершили главную ошибку. Я, по крайней мере.

И дело совсем не в несправедливости (никто ведь нас силой не заставлял на это соглашаться, всегда можно было откланяться и уйти), не в потере потенциальных прибылей и не в утрате контроля над проектом (все проектные решения в рамках «генеральной линии» мы все время принимали сами). Самым существенным моментом была *утрата мотивации.* Еще не начав работу, мы оказывались владельцами исчезающе малой части проекта. *Нашего собственного* проекта, между прочим! По сути, мы изначально превращались в обычных наемных работников, заинтересованность которых в конечном успехе всего дела становилась минимальной, и потому наше усердие зависело только от текущего размера зарплаты... Даже если предположить, что в будущем проект даст какую-то отдачу, наша доля по-любому будет минимальной. В сочетании с моим скептицизмом относительно предлагаемых идей, о которых я писал выше, это в итоге привело к глубинному неверию в достижимость конечного результата[4].

К тому же теперь проект становился даже и формально не нашим: другой аспект соглашения предусматривал передачу всех исходных текстов компилятора вновь создаваемой фирме.

Тут я вижу, как читатель делает характерный жест вроде кручения указательного пальца у виска. Друзья, не судите строго! Как по-другому мы могли ими распорядиться? Начинать работу, зафиксировав наши отдельные права на исходники компилятора? В чем тогда смысл создания фирмы, если она не имеет прав на центральную часть создаваемого ею продукта? Так что мы с Сашей рассматривали передачу компилятора в фирму как простое следствие общей схемы: инвестор вкладывает в фирму материальные средства, а мы — наш интеллектуальный капитал. Другое дело, что можно было (надо было!) ожесточенно и аргументировано торговаться насчет процентов ему и нам и добиться более справедливого распределения. Но если уж мы согласились на 80/20 — что говорить об исходниках? Снявши голову, по волосам не плачут...

Дальше. Те 20%, которые составляли нашу долю, распределялись между пятью участниками, причем нам с Сашей — авторам компилятора — достались ровно такие же проценты, что и трем «новым», которые, очевидно, никак не участвовали в создании компилятора и были, по сути, приглашены в фирму нами. Конечно, эти очень малые проценты не играли абсолютно никакой роли, и даже если бы у нас с Сашей было на процент-другой больше,— реально это бы ни на что не повлияло. Другое дело, что так можно было хотя бы символически обозначить наш преимущественный вклад в проект и тем самым получить хотя бы моральное удовлетворение.

Такое распределение «нашей» части акций — целиком на моей совести: инвестор никак не вмешивался в дележ этих 20%. Мне же просто не хотелось обидеть «новых» участников меньшей долей. Но важнее было другое: предполагалось ведь, что и их вклад в итоговый продукт тоже будет достаточно весомым. Все-таки написать визуальную часть системы программирования и сделать виртуальную машину Си++ — это тоже изрядная работа, так что равное распределение долей казалось своего рода обеспечением их будущего вклада.

Наконец, последнее: должности. Договорились, что генеральным директором новой фирмы будет один из «приглашенных» участников, а техническим директором — я. Такое решение было с моей стороны вполне осознанным, и сейчас я, в общем, склонен считать, что оно было правильным. Совершенно не улыбалось заниматься формально-бюрократическими делами, связанными с регистрацией фирмы, визитами в официальные присутствия и прочей деятельностью, не касающейся непосредственной разработки. А «технический директор» — это как раз то, чем я и должен заниматься: разработка проектных решений, планирование и контроль выполнения работ, вопросы интеграции компонентов системы, ну и, конечно, программирование.

Само по себе решение-то, может, и было правильным... Другое дело, что тогда мы с Сашей, по сути, очень плохо представляли себе деловые качества наших новых коллег и, в частности, того, кто получил позицию гендиректора. Наши мозговые штурмы в проекте Wednesday показали вполне высокий уровень их профессионализма в вопросах проектирования и разработки, кроме того, за ними стоял реальный завершенный и продаваемый продукт — полная система программирования на Си (у нас же с Сашей никакого *продукта,* строго говоря, не было). Однако их деловые качества

[4] Думая об этом сейчас, через много лет после описываемых событий, я удивляюсь: неужели сам инвестор не понимал, что своими 80% он сразу же напрочь убивает нашу мотивацию?! И почему я сам, почувствовав это, не высказал такое опасение?

никак не могли проявиться в наших еженедельных встречах, и потому выбор гендиректора был в чистом виде вопросом доверия к нашим новым партнерам. (Ох, как не хочется писать о неприятном... Но уж взялся за гуж — не говори, что не дюж. Дальше об этом немного будет.)

Саша быстро устранился от процесса переговоров, передоверив наши общие «полномочия» мне,— ему было более чем достаточно получать неплохие деньги за работу, которой до недавнего времени он занимался практически бесплатно, а детали, не связанные с языком Си++ и с программированием компилятора, его не слишком интересовали. С одной стороны, эта «передача полномочий» была, конечно, выражением полного доверия с его стороны. Но в результате я, по существу, оказался вынужден принимать важные решения — как технические, так и организационные — в одиночку, причем в той сфере, в которой мой опыт равнялся нулю (дело усугубляло еще и то обстоятельство, что *я сам тогда не понимал* своей несостоятельности во многих вопросах; да и в людях разбирался плохо, чего уж там). В общем, это была именно та ситуация, когда пословица «одна голова хорошо, а две лучше» приобретала совершенно конкретный и предельно актуальный смысл... Следовало, конечно, все решать вдвоем с Сашей, даже преодолевая его нежелание углубляться в непрограммистские темы. Тем более что к тому времени у меня были возможности убедиться в Сашиной разумности, прозорливости и твердости в вопросах, не относящихся прямо к вопросам программирования.

В общем, вот такой была наша начальная конфигурация. Смейтесь, ругайте, издевайтесь, но мы приняли практически все условия, причем без реальных обсуждений, как должное. Единственное, в чем мы немного поупорствовали, — это размер зарплаты: мы настояли на суммах «немного выше рынка».

Инвестор бурно протестовал, аргументируя это фразой: «Как я объясню своим ребятам, что вы получаете больше, чем они?». Переговорщик он был, безусловно, опытный и искушенный — впоследствии у меня было много случаев в этом убедиться,— но тут он, конечно, классическим образом подставился. Новый гендиректор немедленно отреагировал классическим же образом: «Отношения со своими ребятами — это твои проблемы. Не перекладывай их на нас».[5]

На самом деле, как вы понимаете, ничего хорошего в этом не было. То есть, получать «выше рынка» — это хорошо, кто же спорит. Очень плохо, что это был единственный «успех» в наших переговорах. Все же другие условия, по существу, нами были приняты даже без обсуждений. Иными словами, получилось так, будто нас приняли на работу в фирму, которую основал и которой владеет кто-то другой, и мы согласились на это, удовлетворившись интересной работой и хорошей зарплатой...

Мы определили некую конечную Цель работы нашей новоиспеченной фирмы, и я написал План нашего движения к ней.

Поехали!

В итоге многочисленных обсуждений, предлагая и отбрасывая варианты, прикидывая силы и ресурсы, мы определили некую конечную Цель работы нашей новоиспеченной фирмы, и я написал План нашего движения к ней. Основной срок был два года — почти такой же, как в ситуации начала работ с бельгийцами четыре года назад. Но тогда план, честно говоря, составлялся весьма умозрительно (по-другому было и невозможно: опыта подобных проектов ни у кого не было, и работа начиналась «с нуля»). Сейчас же двухлетний срок появился у меня не «с потолка», а как результат довольно тщательного, как мне казалось, анализа того, что у нас уже сделано, что предстоит сделать, и какими ресурсами мы располагаем.

Сейчас у нас был вполне работоспособный компилятор. Конечно, его надо было дорабатывать, тестировать, интегрировать в будущую среду программирования, — но все равно ситуация качественно отличалась от той, когда у нас не было вообще ничего. А сделать первую работоспособную версию интегрированной среды силами двух разработчиков, у которых уже вроде бы был опыт создания аналогичной системы для Си, — срок в два года казался вполне достаточным.

Так что мне и сейчас кажется, что план сам по себе был правилен, а сроки реальны. Конечно, некоторый свойственный мне оптимизм в этом отношении присутствовал (известно же: чтобы получить настоящую оценку, надо выверенные и обоснованные сроки умножать на число «пи»), но в целом, повторяю, план казался мне выполнимым. Другое дело, что... новых ребят-то я толком не знал и «по умолчанию» предполагал, что они такие же энтузиасты, как и мы с Сашей, — и планировал, исходя из этого неявного допущения.

Так или иначе, наш гипотетический проект Wednesday, казалось, обретал черты реального. И в какой-то момент я искренне поверил, что реализовать его в полном объеме и в заявленный срок нам пятерым вполне по силам! Возник даже некоторый кураж. И первоначальные идеи инвестора насчет «коробочного продукта» и локализации компилятора теперь представлялись не столь уж нелепыми. По мере разработки проектных решений становилось все понятнее, как эти идеи могут быть реализованы, я все ближе к сердцу принимал то, о чем раньше всерьез и подумать не мог...

По этим причинам я был вполне искренен, когда обосновывал и отстаивал

[5] Как мне сейчас кажется (машу кулаками после драки), если бы мы тогда проявили подобную твердость и в других вопросах, то не исключена вероятность, что нам удалось бы настоять на более справедливых условиях.

основные идеи проекта в статье, опубликованной спустя некоторое время в четвертом номере журнала «Мир ПК» за 1999 г. Статья в полном варианте (в журнале ее очень сильно сократили) довольно большая, так что здесь я очень кратко сформулирую только основные идеи.

Русские «плюсы»

Если коротко, то цель нашей компании — создание *взаимосвязанного комплекса программных средств, учебных материалов и методик, направленных на поддержку процесса освоения языка Си++.*

Каковы же основные черты такого комплекса? В целом он, очевидно, должен представлять собой то, что традиционно называется «системой программирования на языке Си++». В него будут входить компоненты, которые всегда присутствуют в системах такого рода: компилятор, библиотеки, пользовательская среда с редактором, отладчиком, менеджером проектов и т. д. Однако ориентация на цели обучения предопределяет ряд особенностей, которые несколько отличаются от привычных.

Первое — оперативная помощь. В системе, ориентированной на обучение, удельный вес этого компонента должен быть, очевидно, существенно бóльшим, нежели в обычных системах. Это относится как к объему помощи, так и к ее направленности. Например, каждое диагностическое сообщение компилятора должно подробно объясняться; для всех случаев появления сообщения должны быть не формальные (как у многих известных систем), а содержательные примеры. Конструкции языка, библиотечные функции, приемы и «правильный» стиль программирования — все должно подробно объясняться, на все должны быть примеры и все должно быть легко доступно. Такая же обширная помощь должна быть, разумеется, и по самой системе (меню, команды и т. д.).

Второе — специальные прикладные библиотеки, ориентированные именно на освоение языка. Такие библиотеки (например, двумерная графика, простые средства создания графического пользовательского интерфейса, математические вычисления и т. п.) должны помочь начинающему программисту как освоить объектно-ориентированный подход, так и приобрести первые навыки работы с библиотечными окружениями. Такие библиотеки могли бы, помимо прочего, играть роль своеобразного «переходника» между стандартными для некоторого операционного окружения библиотеками (например, Windows API) и программой начинающего программиста. В противном случае ему неизбежно придется одновременно осваивать две очень непростые вещи — собственно язык Си++ и системные библиотеки, которые свяжут его программу с конкретной средой.

Третий аспект заключается в особенностях выполнения программ на языке Си++. Хорошо известно, что язык Си++ (как и его предшественник — Си) — незащищенный. Иными словами, возможность возникновения некоторых семантически некорректных ситуаций в них «заложена» изначально, и такие ситуации зачастую не контролируются ни при компиляции, ни при исполнении. Примерами могут служить выход индекса за границы массива, работа с неинициализированными данными, обращение к несуществующему динамическому объекту и т. д. Наличие в системе развитого отладчика исправляет ситуацию лишь частично. Если же говорить о начинающем программисте, то выявить подобные ситуации и, что важнее, научиться в дальнейшем их не допускать будет для него очень непросто. В то же время можно предположить, что быстродействие результирующей программы — не критический фактор в такой системе, хотя бы потому, что в большинстве случаев программы будут сравнительно небольшими.

Каждое диагностическое сообщение компилятора должно подробно объясняться; для всех случаев должны быть не формальные а содержательные примеры.

По этим причинам в системе, ориентированной на обучение, была бы оправданной не вполне традиционная схема выполнения программ на Си++. Компилятор, производя полный комплекс семантических проверок (согласно Стандарту Си++), генерировал бы не выполнимый объектный код, а некоторое промежуточное представление (ПП), которое содержит всю информацию об исходной программе. А выполнение программы могло бы быть реализовано как *интерпретация* этого промежуточного представления с помощью специального компонента, который можно назвать *виртуальной машиной* Си++.

Описанная схема давно и хорошо известна. Недавние технологические решения, связанные с языком Java и его виртуальной машиной, показывают жизнеспособность и достоинства этого подхода, хотя и недостатки тоже вполне очевидны. Однако сейчас хотелось бы не останавливаться на обсуждении этих вопросов, а отметить преимущества такой схемы в контексте заявленных целей. Разумеется, интерпретационная схема исполнения хороша не потому, что ее проще реализовать (сложность семантики языка Си++ такова, что сделать для него генератор кода едва ли не легче, чем хорошую виртуальную машину). Принципиальный выигрыш заключается в том, что именно такой, полностью контролируемый режим исполнения может обеспечить *диагностический сервис,* необходимый для начинающего программиста. С помощью виртуальной машины можно выполнять такие динамические проверки, которые весьма затруднительны или вообще невозможны при выполнении сгенерированного объектного кода.

Четвертое — в такой системе необходим традиционный учебник по языку Си++. При этом было бы крайне желательно, чтобы, во-первых, такой учебник был написан людьми, хорошо

понимающими особенности целевой (русскоязычной) аудитории и, во-вторых, был внутренне связан с самой программной системой. Он должен быть компактным (устрашающие объемы современных книг по программированию сами по себе способны отпугнуть начинающих) и в то же время объяснять все основные свойства языка. Далее, такая книга не может быть просто добросовестным описанием языка. Она обязательно должна нести определенный методический заряд, помогающий читателю увидеть за непривычными конструкциями обоснованные технологические подходы и методы, дисциплинирующий и не позволяющий ему свалиться в расхлябанный «хакерский» стиль программирования, столь характерный, к сожалению, для многих. Вывод из сказанного один: было бы правильно, если такой учебник был написан либо авторами самой программной системы, либо в тесном сотрудничестве с ними.

И, наконец, *последнее:* входной язык и компилятор. Есть искушение в простой логике: если система создается для обучения, то нет смысла реализовывать в компиляторе все возможности языка, тем более что многие из них (например, шаблоны) как будто специально спроектированы так, чтобы сделать их реализацию буквально программистским подвигом. Если следовать такой логике, то встает вопрос: что выбросить? И тут оказывается, что если задумываться о правильном стиле программирования на Си++, о том, что называется современной парадигмой программирования, и с этой точки зрения рассматривать те или иные свойства языка, то, строго говоря, выбросить *ничего нельзя.* Например, шаблоны являются важнейшим инструментом обобщенного программирования — мощного и многообещающего подхода, одного из наиболее заметных достижений науки программирования последних лет; к тому же все стандартные библиотеки основаны на них. Выброси из компилятора реализацию шаблонов — и учебник по языку тут же превратится в ординарный пересказ первых изданий Страуструпа, Шилдта и многих других авторов. Откажись от поддержки исключительных ситуаций — и станет непонятно, как донести до пользователя принципы безопасного программирования.

Таким образом, единственно возможной альтернативой представляется точное следование Стандарту языка Си++ в полном объеме, сколь бы тяжелой эта задача ни казалась.

На этом пути есть и еще варианты: зачем реализовывать систему целиком? Почему не воспользоваться тем, что уже есть? Ведь технически можно, например, встроить в свою систему компилятор любого известного изготовителя, организовав передачу ему исходных текстов и перехват диагностических сообщений. Или, наоборот, включить свой компилятор в «фирменную» среду разработки — многие известные системы допускают подобное. Возможен и такой относительно простой путь, как создание не полноценного компилятора, а препроцессора в более простой язык Си, вроде известного cfront...

Однако внимательный анализ показывает, что ни один из подобных путей не подходит. Такой «полуфабрикатный» продукт будет неизбежно привязан к включаемой (или включающей) системе, и в случае смены версии последней ни одна фирма не даст гарантию ее совместимости с нашими компонентами.

Начало: планы, персоны и обязанности

Первым приобретением новоиспеченной компании («новоиспеченной» — это громко сказано, кроме названия ничего не было, она даже не была зарегистрирована) были новые черные офисные столы и черные же кресла. В общем-то, вполне скромные и без изысков, тогда они казались нам верхом элегантности и осязаемым символом нашего будущего успеха. Мы разместили эту мебель в нашей комнате в университете — там же, где мы обсуждали еще недавно проект Wednesday, не предполагая, что совсем скоро наши планы начнут воплощаться в жизнь... Раньше мы сидели в этой комнате вдвоем с Сашей (и иногда студенты приходили), но ее размер оказался вполне достаточным, чтобы вместить еще троих «новеньких». Через несколько месяцев, правда, инвестор посчитал, что будет правильнее, чтобы фирма располагалась не на «ничейной» территории, а в том же офисе, где находилась другая его компания и где сидел он сам, — и мы со всем имуществом переехали на Дмитровское шоссе.

Конечно, мы купили и компьютеры — не слишком навороченные, но вполне подходящие для наших задач. Они верой и правдой прослужили нам все время, а посетив фирму через несколько лет, я с ностальгией увидел там несколько «своих» системных блоков и мониторов — совершенно устаревших к тому времени, но тем не менее исправно работающих.

В соответствии с Планом, наша работа была организована так. Мы с Сашей занимались доработкой компилятора: продолжали тестировать его, исправляли ошибки, дописывали код, реализующий последние нововведения в Стандарт, — прежде всего это касалось механизма шаблонов и деталей семантики виртуальных функций и виртуальных базовых классов.

Кроме того, мы заново проектировали интерфейсы компилятора для включения в интегрированную среду: теперь компилятор должен был выступать не только в роли независимой утилиты, вызываемой из командной строки, но и служить частью системы программирования. Помимо этого, нужно было сделать внутренние таблицы компилятора доступными извне. На самом деле это была нетривиальная задача, и дело было не просто в архитектурных изменениях программы: следовало переделать (а частично написать заново) и свести в единый интерфейсный компонент все способы и техники доступа к таблицам, дереву и типам. Этот интерфейс изолировал семантическое представление от других компонентов компилятора, тем самым оно становилось независимым компонентом. Такое построение сообщало всей архитектуре определенную инновационность и позволяло строить на ее основе

различные языково-ориентированные инструменты, а не только (как обычно) генераторы объектного кода.

Программная поддержка локализации компилятора (русскоязычные идентификаторы и русские варианты служебных слов), как и ожидалось, не потребовала значительных усилий и была сделана довольно быстро (параллельно с другими работами) — во всяком случае, значительно быстрее, нежели были придуманы и согласованы сами русские эквиваленты служебных слов.

С прохождением тестов все было тоже более или менее в порядке (хотя довести процент успешных тестов до 99 удалось только к концу описываемого периода).

Я сделал таблицу, в которую регулярно заносил информацию о пройденных и непройденных тестах, и периодически вывешивал ее свежий вариант (в виде гистограммы) около монитора. Текущее состояние тестирования и общий прогресс были видны очень наглядно.

(А много лет спустя, при тестировании компилятора Зоннон, эта технология подверглась глубокой модернизации: Роман Митин написал PHP-программу, которая автоматически извлекала из логов тестирования нужную информацию, сравнивала ее с данными предыдущих прогонов и выводила на наш Web-сайт последние результаты в наглядном виде и в сравнении с предыдущими.)

Основные же проблемы возникли в связи с компиляцией реальных программ. Самой важной из таких программ была, что вполне понятно, реализация *стандартной библиотеки* Си++ — точнее, тех компонентов, без которых обойтись было невозможно. Наибольшие трудности доставила STL (Standard Template Library). Первоначально это была независимая библиотека, разработанная Александром Степановым, она воплощала фундаментальные принципы введенной им концепции обобщенного программирования (generic programming). В 1994 г. STL стала официальной частью библиотеки Си++ и в таком качестве вошла в Стандарт. Мы взяли свободную реализацию этой библиотеки от Плоджера[6] и «проталкивали» ее исходники через наш компилятор. Это была непростая и довольно долгая работа, так как библиотека эта, если можно так сказать «насквозь шаблонизирована»: там активно используются, кажется, все существующие возможности и особенности механизма шаблонов. Полностью эта работа была завершена только через два года.

Предполагалось, что гендиректор со своим товарищем займутся интегрированной средой и, в частности, онлайновой помощью. Мне показалось, что у них есть вполне достаточный опыт по вопросу реализации подобных сред; во всяком случае, их система программирования для Си выглядела очень профессионально. Я, правда, не интересовался степенью их личного вклада в разработку этой среды (и, как выяснилось, напрасно!), но судя по тому, что они охотно согласились программировать аналогичную среду для Си++, подобная работа была им не в новинку.

В нашей системе мы хотели сделать гораздо более всеобъемлющую и более тщательно проработанную систему помощи, нежели в широко распространенных тогда аналогичных средах Microsoft и Borland. Помимо обычной справочной информации, мы собирались включить туда стандарт Си++ целиком, будущий учебник по языку (который будет либо написан нами самими, либо заказан у кого-то из сторонних авторов), развернутую справку по диагностикам компилятора и многое другое.

Наконец, третий из «новичков» должен был заняться виртуальной машиной Си++ и линковщиком промежуточного представления.

Забегая вперед, скажу, что третий разработчик оказался единственным из наших новых коллег, кто в целом успешно справился со своей частью проекта: виртуальная машина была сделана и показала работоспособность (на тестах; признаться, не помню, дошло ли дело до ее испытания на реальных программах до того, как «все кончилось»...). Но это, к сожалению, не помогло...

О виртуальной машине я уже писал, а о линкере стоит сказать несколько слов. Одна из известных проблем, связанных с «негативным» наследием языка Си, которое Си++ пришлось воспринять, — проблема объединения объектных кодов от нескольких единиц компиляции и связывания объектов в этих кодах. Эта проблема прямо идет от требования совместимости, которое при проектировании Си++ полагалась безусловным. А один из непременных аспектов требования совместимости — это условие, согласно которому стандартные «сишные» инструменты и, в частности, редакторы связей (линкеры) должны «гладко» линковать объектные коды, полученные как из-под Си-, так и из-под С++-компиляторов.

Здесь можно долго обсуждать различные аспекты проблемы линковки (об этом написаны книги), поэтому ограничусь одним примером, непосредственно связанным с шаблонами Си++. Пример очень простой: пусть имеется некоторый шаблон класса, который посредством директивы include препроцессора (еще один откровенный архаизм обоих языков) включен в несколько единиц компиляции. И в каждой такой единице он настраивается с получением класса-по-шаблону (или, согласно последней принятой терминологии, с получением специализации класса).

Так вот, если предположить, что в двух различных единицах компиляции шаблон настраивается *одним и тем же типом*, то понятно, что и специализации получаются идентичными. В то же время каждая единица трансляции компилируется без учета общего контекста: компилятор просто не знает, с какой единицей трансляции будет впоследствии линковаться код, полученный из данной единицы трансляции (более того, этого зачастую не знает и программист, написавший эту единицу!). Поэтому в обеих единицах остается совершенно одинаковый код, полученный из двух настроек одного и того же шаблона.

[6] Web-сайт: *http://www.dinkumware.com/*. Книга: The C++ Standard Template Library, P.J. Plauger, Alexander Stepanov, Meng Lee, David R. Musser, ISBN-10: 0134376331, ISBN-13: 978-0134376332.

По логике вещей, при линковке хорошо бы оставить в результирующем коде только один экземпляр настройки шаблона. Но... существующие линкеры работают на слишком низком уровне, ничего не знают не только о шаблонах, но и о классах (линкеры же «сишные»), и потому не в состоянии обнаружить повторяющийся код. Эта проблема давно известна и получила название code bloat («разрастание кода»).

А наш компилятор порождает не низкоуровневый объектный код, а более «умное» промежуточное представление, в котором сохранена вся информация об исходной программе и, в частности, о шаблонах и их настройках. Поэтому если реализовать инструмент, который бы связывал вместе промежуточные представления нескольких единиц трансляции, то код-дубликат можно было бы довольно просто идентифицировать и удалить.

Помимо простой редукции лишнего кода, такой линкер мог бы делать и другие полезные вещи, вплоть до весьма продвинутой глобальной оптимизации — опять же благодаря полной информации о программе, сохраненной в промежуточном представлении.

По существу, виртуальная машина и линкер промежуточного представления (ПП) задумывались как первые клиенты промежуточного представления и должны были служить своего рода «подопытными кроликами»: на них предполагалось отладить технологию использования программного интерфейса в ПП и убедиться, что этот интерфейс удобно использовать и потому можно предлагать и для других инструментов-клиентов.

Что говорит компилятор

Казалось бы, реализация выдачи диагностических сообщений — вполне рутинная работа, неотделимая от программирования собственно компилятора. В нашем случае это было совсем не так.

Во-первых, нужно было обеспечить многоязычность диагностик. Даже это требование приводило к необходимости более систематического подхода к такому, вроде бы, тривиальному аспекту реализации. Сделать простую печать сообщения, формируемого динамически, «по месту» появления ошибки, уже не получалось. Каждое диагностическое сообщение теперь представляет собой некоторый шаблон, заготовленный заранее, например «необъявленная функция '%1' в классе '%2'». При выдаче диагностики этот шаблон параметризуется реальными именами (в данном примере литеральным именем необъявленной функции и именем класса) и в таком виде выдается в выходной протокол. Все шаблоны группируются в массив; на самом деле, структура информации еще сложнее, так как для каждого шаблона имеется его русскоязычный эквивалент, и в зависимости от параметра компиляции «язык диагностик» выбирается нужный вариант шаблона.

Но это еще не все. Так как первоначально мы ориентировали проект на поддержку обучения (во всяком случае, это была одна из заявленных нами целей), необходимо было обеспечить дополнительный сервис. Хотелось бы предоставить развернутое объяснение (комментарий) к каждому диагностическому сообщению: что может реально означать такая (по вынужденности краткая) диагностика, каковы наиболее вероятные причины ее возникновения, что следует изменить в программе, чтобы исправить ошибку. Иногда имело смысл сопроводить комментарий типовым примером ситуации, которая может вызвать данную ошибку. В некоторых случаях написать комментарий и пример было несложно, иногда же это требовало заметных усилий и даже определенной фантазии: нужно было представить различные ситуации в программах, которые могли привести к получению данного сообщения.

Разумеется, комментарии тоже должны быть двуязычными.

Наконец, по каждому сообщению необходимо было предоставить ссылки на онлайновый учебник по Си++ (который планировался в составе системы), а также ссылку на «официальный» источник информации — Стандарт языка. Имелось в виду, что по этим ссылкам можно было найти формальные правила, нарушение которых в программе послужило причиной появления данной диагностики.

Ну а теперь учтите, что диагностик в компиляторе *более тысячи* — и представьте, насколько тяжелой была эта работа.

Понятно, что подобный объем информации, связанный с диагностическими возможностями компилятора, следовало подходящим образом структурировать и обеспечить удобный процесс работы с ней: добавлять и изменять шаблоны, комментарии и т. д. Надо еще учесть, что использоваться вся эта информация могла по-разному. С одной стороны, шаблоны самих диагностик должны выдаваться компилятором и потому входить в его исходный текст. С другой, те же диагностики вместе с комментариями должны служить частью документации по компилятору (а проработав тринадцать лет в «ящике», я хорошо представлял себе требования больших компаний и госорганизаций в отношении документации). Наконец, и диагностики, и комментарии, и ссылки на стандарт должны были войти составной частью в оперативную помощь по компилятору: мы же собирались делать интегрированную среду программирования, не забывайте! Так что нужно было одновременно предоставить удобный способ добавления и правки диагностической информации и, с другой стороны, обеспечить генерацию выходных данных в различных форматах. К тому же весь этот сервис должен быть не слишком сложным в сопровождении и не содержать экзотических и «тяжелых» компонентов.

Сделали так. Вся диагностическая «база данных» представляла один большой файл (документ) в формате Word. Содержательно это была сложная таблица, в ячейках которой и содержалась вся описанная информация. Один из молодых ребят, которых мы взяли в фирму через некоторое время, написал VBA-приложение (по-нынешнему «плагин») для редактора Word, который извлекал из этой таблицы нужную информацию и генерировал текст в подходящем формате.

Таким образом, вся диагностическая информация была сосредоточена в одном месте. Когда нужно было с ней работать, мы пользовались удобным и мощным редактором, а итоговые массивы данных для различных компонентов системы генерировались автоматически.

Конечно, описанный способ организации данных — не единственно возможный. Для полноты картины можно добавить, что через некоторое время, уже после нашего ухода, «база данных» была переведена в формат XML, а вместо плагина-генератора был написан XSLT-преобразователь, который и генерировал выходные тексты в нужных форматах.

Как организовать работу: инструменты и технологии

Когда мы работали над компилятором вдвоем с Сашей, то, в общем, не испытывали особых потребностей в какомлибо организационном упорядочении работы, кроме уж самых насущных вещей вроде системы ведения версий. Файлы с компилятором лежали у нас на SPARC'е в старенькой системе CVS. Саша работал под спарковским UNIX, я — под Windows. Тексты редактировали, соответственно, vim и Multi-Edit[7], отлаживали компилятор одним и тем же отладчиком gdb, который успешно работал на обеих платформах. Нужные для работы файлы я, как полагается, забирал из CVS, переписывал по локальной сети на свою персоналку и компилировал посредством либо gcc, либо компилятора Borland, а через некоторое время стал использовать еще и Microsoft Си++. Тестировался компилятор (регулярно, ночами) на SPARC, периодически тесты прогонялись и на персоналках.

Все «общие» проблемы в компиляторе, которые возникали на стыке «моих» и «его» компонентов, обсуждались и решались устно (при том, что сама принадлежность тех или иных компонентов конкретному разработчику была весьма условной). Если нужно было поработать с «чужим» файлом, то один из нас возвращал его в базу CVS, а другой забирал его оттуда — это тоже решалось «на месте», так как сидели мы рядом.

Вообще, в процессе той работы — настолько сильно она забирала все внимание — я, честно говоря, не сильно интересовался какими-то не относящимися к ней вещами. В частности, способы «правильной» организации совместных работ и правила построения команд программистов меня, можно сказать, совсем не интересовали. Конечно, полагающиеся книги по этой проблематике я в свое время прочитал и соответствующие термины знал, однако на практике применять методики и инструменты для организации этой самой совместной работы нам двоим было совершенно ни к чему.

В новой фирме все изменилось. Во-первых, нас просто стало больше — и уже по одному этому требовалась определенная, пусть неформальная, но достаточно четкая дисциплина взаимодействия. Во-вторых, теперь в наших планах был не один компилятор, а несколько продуктов, которые по смыслу должны быть очень тесно друг с другом связаны. В-третьих, каждый из этих продуктов необходимо было развивать в нескольких аспектах. Например, компилятор нужно было дорабатывать для включения в среду, добавлять продвинутые средства диагностики, русифицировать, выделять из него компонент доступа к промежуточному представлению — и, кроме того,

[7] Замечательный был редактор! Мощный, настраиваемый, со своим внутренним языком, на котором можно было перепрограммировать всю его визуальную часть — не говоря о том, что в него удобно интегрировались многие компиляторы командной строки. В свое время я даже книжку о нем написал. Вскоре подобных редакторов появилось довольно много, однако долгое время он считался, пожалуй, лучшим в своем классе.

продолжать его тестировать! И делать это должны были (по крайней мере, так планировалось) не только мы с Сашей.

Дальше. Быстро выяснилось, что нам нужна поддержка документооборота. Такая система понадобилась не из абстрактных бюрократических соображений («чтобы на любой чих был документик») — прежде всего нам реально нужно было фиксировать общие технические решения, которые служили основой совместной работы. Решения вырабатывались в процессе обсуждений, и было бы очень полезно, чтобы сохранялись не только сами решения, но и альтернативы, и отвергнутые варианты. Более того, решения часто корректировались в процессе реализации, и нужно было вносить в них изменения, поддерживая их в актуальном состоянии, а также донести внесенные изменения до тех, на чью работу они влияли.

Раньше у нас был, по существу, единственный рабочий документ — проект, который мы в начале работы сделали для бельгийцев, плюс кое-какие фирменные руководства (правила кодирования, описание формата «обобщенного ассемблера» и т. п.). Оперативное взаимодействие с заказчиками шло посредством электронной почты, и в каких-либо системах поддержки процесса ни мы, ни бельгийцы нужды не испытывали.

Теперь же у нас стало появляться много различного рода документов — от общих и детальных планов до всякого рода проектных материалов, с которыми должны были работать несколько человек. Возникали и презентации, и рекламные тексты, и статьи для журналов. Некоторые электронные письма, да и внутренняя переписка тоже были достаточно важными, чтобы их сохранять. Уже во время наших встреч по проекту Wednesday мы начали вести протоколы обсуждений, в которых фиксировали важнейшие проектные решения. Тогда это происходило немного «понарошку» (мы и не представляли, что от наших планов до реальности окажется такое небольшое расстояние), однако сейчас такие протоколы приобрели характер необходимости, они становились реальными рабочими материалами.

Кроме того, в процессе работы между участниками возникало очень много «рабочих», текущих проблем, требовавших, помимо быстрого разрешения, еще и фиксации — с тем, чтобы впоследствии можно было вернуться к истории вопроса и увидеть, как данная проблема была решена.

В качестве инструмента совместной работы и управления всей проектной информацией нашими новыми коллегами был предложен Lotus Notes. Сначала он показался нам довольно громоздким и неуклюжим инструментом, который к тому же был на удивление плохо спроектирован и реализован. До этого времени практически все западные продукты казались мне если не верхом совершенства (неудобств хватало и у Borland, ныне Embarcadero, и у Microsoft), то все-таки весьма грамотно сделанными вещами, и я уж никак не предполагал, что «у них», оказывается, на продажу можно выставлять настолько неаккуратно разработанные программы. Это наблюдение показалось мне тогда почти откровением.

В новой фирме все изменилось — и уже это требовало строгой дисциплины взаимодействия.

У Lotus Notes оказался очень неудобный редактор — у него отсутствовало много свойств, которые после опыта работы в Word казались обязательными «по умолчанию», у него был, на мой взгляд, неграмотно сделанный и неочевидный интерфейс, к тому же принцип WYSIWYG не был выдержан полностью. Импортирование документов других форматов если и поддерживалось, то настолько ограниченно и неудобно, что о Word пришлось забыть.

Методика совместной работы, диктуемая идеологией Lotus Notes, была невероятно громоздкой, «бюрократической» и в наших условиях практически неработоспособной. Возможно, для очень больших коллективов обилие правил, обязательных процедур, регуляций и ограничений было оправданным, но не для таких компактных команд, как наша. К счастью, система оказалась достаточно гибкой, и один из ребят (который был горячим сторонником ее использования и убедил нас взять именно ее) настроил ее на более простую и понятную модель взаимодействия, которую мы с Сашей, скрепя сердце, приняли.

Интерфейс Lotus Notes — как в настольном, так и в Web-варианте, — кажется, был верхом нелепости и несовершенства: такое впечатление, что его слепил студент, причем не в порядке сторонней «подработки», а в качестве очередной курсовой работы, торопясь закончить ее к определенному сроку — под угрозой «неуда». Если говорить о Web-интерфейсе, то даже при тогдашних ограниченных выразительных возможностях HTML, даже без использования CSS и JavaScript, любой толковый программист — даже тот, кто никогда прежде не сталкивался с подобными задачами,— мог бы сделать этот интерфейс куда функциональнее, удобнее и элегантнее...

В целом, переход на регулярное использование Lotus Notes в ежедневной работе дал нам серьезный опыт по крайней мере в двух аспектах: мы вполне осознали полезность и необходимость инструментария совместной работы и, кроме того, благостная прежде картина всеобщего высокого качества широко рекламируемых западных продуктов была несколько размыта и тем самым приобрела больший объем и полноту.

Мы полностью «переехали» на платформу Windows; бельгийские SPARC остались в университете. Теперь весь код мы писали и отлаживали с использованием микрософтовского Си++ (не помню, когда у его названия появилась приставка Visual — уже тогда или позже), а то, что называется production code, каждую пятницу формировалось на сервере с помощью Visual Age for C++ —

другого продукта IBM. Там же проходили регулярные прогоны тестового пакета под управлением специального монитора тестирования, который написал один из молодых ребят.

В отличии от Lotus Notes, этот Visual Age был без преувеличения выдающейся системой — с потрясающе красивым дизайном, очень качественной реализацией Си++ (нововведения в стандарт Си++ там были реализованы в большем объеме, нежели у Microsoft), передовыми и даже уникальными средствами линковки и сборки программ. Правда, система была исключительно жадная до ресурсов компьютера (кажется, в сервер пришлось даже ставить дополнительную память только для того, чтобы Visual Age мог хоть как-то работать) и безумно медленная — по-моему, только поэтому мы не перешли на повседневное ее использование.

В общем, замечательный был продукт этот Visual Age for C++. Говорю «был», поскольку сейчас его уже нет. В свое время казалось, что этот монстр — на века, и ничто вроде бы «не предвещало»... К тому же никак не верилось, что такой гигант, как IBM, может «слить» свой продукт, да еще такой выдающийся (они вон до сих пор и Кобол, и PL/1 поддерживают). Но через какое-то время IBM объявила, что Visual Age больше развиваться и поддерживаться не будет — уж не помню, какие причины выдвигались. Это решение сильно ударило по фирме: к тому времени несколько программ, ею разрабатываемых, оказались сильно «заточены» под эйджевские библиотеки визуализации, аналогов которым не было. Знаю, что выходили ребята из этой ситуации довольно долго...

«Команда»

Довольно быстро стало понятно, что наш гендиректор любит всевозможные «оформительские» аспекты, обсуждения и разговоры гораздо больше, чем реальную работу над проектом.

Он проявлял редкую старательность и истинную креативность при подготовке всевозможных внутрифирменных бумаг: его рекламные проспекты, отчеты для инвестора, протоколы обсуждений и даже зарплатные ведомости являли собой аккуратно и любовно сотворенные чудеса оформительского искусства. Он увлеченно изучал Microsoft Project, при помощи которого рисовал очень красивые схемы и графики нашей работы с взаимосвязями между проектами, с датами этапов... Он очень любил заниматься покупками для фирмы, тщательно выбирая стулья, канцтовары и писчебумажные принадлежности. Он постоянно ездил во всевозможные присутственные места (при этом умудрился за два года так и не зарегистрировать нашу фирму!), постоянно носил с собой пейджер, по которому получал многочисленные сообщения[8]. Он произносил правильные слова, рассказывая об особенностях нашей будущей системы, дал вместе со мной несколько интервью для компьютерных журналов и участвовал в полемике по поводу нашей реализации Си++ на каком-то электронном форуме...

Поймите меня правильно: его деятельность была, безусловно, необходима фирме (в самом деле, ну как же обойтись без красивых отчетов и бумаги для принтера?). Однако нас было всего пятеро, и рабочий план никак не предусматривал, что один из нас будет целиком занят подобной работой! — мы же не контора на сто сотрудников, которой необходим *специальный* человек, обеспечивающий бесперебойное снабжение канцтоварами и каждый месяц привозящий из банка зарплату! А чем меньше команда, тем серьезнее общий результат зависит от работы и вклада каждого. Получилось же, что у нас только четверо разработчиков и один начальник (он же снабженец)...

Уже, наверное, не стоит уточнять, что за все два года, пока шла наша совместная работа, гендиректор не написал, по-моему, ни одной строчки кода. Единственный его реальный результат за это время — прототип подсистемы оперативной помощи, состоящий из двух или трех HTML-файлов, заполненных заготовками текстов (то, что эти заготовки были написаны не им, и так понятно).

Слишком поздно я осознал простую вещь: он *вообще не программист*. Умение разговаривать (вполне, впрочем, умно и содержательно) на «наши» темы вовсе не обязательно предполагает способность воплощать обсуждения в реальные программы или хотя бы в грамотные технические проекты. В такой оценке нет никакой негативной окраски — не всем же писать программы! Но фирме-то он нужен был именно как программист, и мой начальный план составлялся в расчете на его прямое участие в разработке. Титул «генеральный директор» был, скорее, данью традиции и подразумевал не «освобожденное» руководство всем и вся, а некоторую (по идее, очень небольшую) долю дополнительных обязанностей.

В общем, «минус один». И для команды в целом, и для моей персональной «кармы». Для меня даже «минус два»: один минус за неумение разбираться в людях (должен был вовремя, еще до описываемых событий, понять, кто есть кто среди наших новых коллег), а второй — за скрытое удовлетворение тем, что можно было абстрагироваться от всевозможных неинтересных обязанностей, коли нашелся человек, готовый ими заниматься.

...Почему он все-таки так часто ездил во всякого рода присутственные места? Почему он проводил там так много времени? *Что, собственно, он там делал?* Это для меня до сих пор загадка...

Товарищ нашего гендиректора, напротив, был настоящим программистом. Он знал и умел многое; некоторые вещи (например, тонкие особенности использования ООП и реальные примеры того, что несколько позже стало называться design patterns) я впервые узнал от него. Именно он установил и поддерживал всю инфраструктуру совместной работы, придумал простую модель взаимодействия, «продавил» наш консерватизм, буквально заставив

[8] Мобильные телефоны тогда только-только входили в моду и были очень дорогими. Нисколько не сомневаюсь, что сейчас он целыми днями не отнимает трубку от уха...

пользоваться «фичами» Lotus и «тасками» в повседневной работе.

Наконец, это прежде всего он сумел добиться, чтобы наш компилятор «скушал»-таки библиотеку STL. Работа по STL заняла довольно долгое время — гораздо дольше, чем планировалось,— и в некоторой степени именно она «задвинула» в итоге проект графического интерфейса. Впрочем, понятно, что он и не смог бы в одиночку (без гендиректора) сделать хотя бы прототип такого интерфейса...

Между прочим, когда я в начале всей эпопеи разрабатывал план работы и писал технический проект, то предполагал сделать его руководителем и ответственным за все направление, связанное с графическим интерфейсом пользователя. Однако будущий гендиректор, прочитав план, поднял меня на смех, заявив, что его товарищ умеет только программировать и совершенно не годится на роль руководителя разработки. Я не мог не принять это мнение в расчет (они же отлично знают друг друга, думал я, им виднее) и поставил ответственным самого гендиректора; на это он не возражал.

Будущее показало, что при принятии подобных решений нельзя полагаться на отдельные консультации, да еще с «заинтересованными лицами». Впоследствии как раз «программист» и стал тем человеком, который, не жалея себя, вытягивал компанию из глубочайшего кризиса, и если и не добился окончательного успеха, то определенно смог переломить негативную тенденцию. А «гендиректор» за два года не сделал ровным счетом ничего ни в разработке, ни в управлении фирмой и в конце концов вообще ушел из сферы ИТ.

Третий участник все время стоял несколько особняком. Вообще-то это своеобразное умение — демонстрировать сильную вовлеченность в дела фирмы, проявлять неподдельный интерес в обсуждениях, браться за достаточно ответственный компонент — виртуальную машину Си++ — и при этом по факту суметь получить для себя «особые» условия... Дело в том, что в самом начале всего дела, когда обговаривались обстоятельства и детали организации новой компании, он заявил, что сразу не может покинуть свою прежнюю фирму, так как должен выполнить некие взятые на себя обязательства. Но, как он всех уверял, это произойдет очень скоро, «как только, так сразу».

Надо ли говорить, что этого «сразу» так и не произошло. Все это время он занимался делами своей фирмы, у нас появлялся раз в неделю на «митингах» и уходить с той работы отказывался. В качестве оправдания каждый раз строились различные по степени ловкости объяснения, сводящиеся к тому, что, мол, я же *работаю*, и что́ изменится, если я буду приходить сюда каждый день, а не раз в неделю?

Оказалось, что парень, по сути, работал на две компании одновременно (получая зарплату в двух местах, между прочим), и всем было понятно, где его *настоящее* место работы. Ситуация вызывала общее недовольство. Саша едва сдерживался от того, чтобы высказать ему все в глаза. Удерживала его, я думаю, лишь природная деликатность, да еще тот факт, что он, хоть и был одним из ключевых разработчиков, без которого вообще ничего бы не состоялось, оставался все-таки самым молодым членом команды... По-хорошему, такое положение дел терпеть было нельзя. Однако гендиректора эта проблема почему-то ничуть не заботила, и в ситуациях явно высказываемого возмущения, которые вскоре стали периодически возникать, занимал подчеркнуто нейтральную позицию, а то и вставал на сторону третьего.

Работа по виртуальной машине начала реально продвигаться только со второго года,— и то больше силами некоего сотрудника «его» фирмы, которому этот третий, по сути, отдал проект на «аутсорсинг» — уж не знаю, на каких условиях.

Описанная ситуация с третьим участником — самая неоднозначная и морально для меня очень некомфортная. Дело в том, что, честно говоря, я до сих пор не понимаю, как следовало относиться к такой явной...

если не нелояльности, то отстраненности по отношению к общему делу. Многие, возможно, скажут, что от таких людей следует избавляться, так как в критических ситуациях положиться на них нельзя. Теоретически это, наверное, правильно... Но парень-то был (и есть) очень квалифицированный, умелый и опытный (он, собственно, и был автором того компилятора Си, который они показывали при нашем знакомстве) — да и сделал же он в итоге эту виртуальную машину!

Начались «маркетинговые» движения. Инвестор «подгонял» нам людей, которые должны были заниматься как бы продвижением наших продуктов.

Очень неприятная ситуация. До сих пор переживаю. Никому не пожелаю попасть в такую. Дал себе зарок: впредь не допускать даже теоретической вероятности возникновения подобных коллизий.

Кризис

Примерно через год после начала работы всем стало понятно, что «коробочный продукт» у нас не получается. Компилятор уже вполне мог считаться в целом готовым, остальные компоненты — стандартные библиотеки (в минимально необходимом объеме) и виртуальная машина если и отставали от компилятора, то все-таки имели шансы быть завершенными к концу заявленного двухлетнего срока. Однако ни графического интерфейса, ни справочной системы не было даже в виде прототипа. К тому времени в фирме, кроме нас, уже работало трое молодых ребят, но даже с их помощью исправить ситуацию было нереально.

Стало ясно, что необходимо менять стратегию. Мой изначальный скептицизм относительно коробочного продукта (который я в свое время задавил в себе, приняв идею инвестора) теперь разделялся всеми участниками, как и убежденность в том, что следует переориентировать разработку в направлении изготовителей системного ПО для спецпроцессоров. Несмотря на то, что в этой сфере безраздельно царил Си, шансы внедриться туда с «плюсами» все-таки были.

Кроме того, оставались, казалось, перспективы и у другого направления — создания на основе нашего ПП отдельных инструментов: статических анализаторов, например. Один такой инструмент у нас должен был вскоре появиться — виртуальная машина Си++. Хотя изначально она задумывалась как часть будущей системы программирования и отладки, но можно было предложить ее покупателям (в основном, конечно, корпоративным) как инструмент анализа динамического поведения программ. Через некоторое время сторонним разработчиком, моим коллегой из МГУ, был сделан программный визуализатор, который, принимая на вход ПП, наглядно и детально, в виде дерева, показывал статическую структуру программ на Си++. Сейчас подобные средства уже вполне обычная вещь, в том или ином виде они присутствуют во многих интегрированных средах, но тогда, в самом начале двухтысячных, это выглядело «круто». Да и сейчас степень детализации программы у этого инструмента выше, чем у многих известных аналогов.

Постепенно начались «маркетинговые» движения. Инвестор периодически «подгонял» нам людей, которые должны были заниматься как бы продвижением наших продуктов на рынок. Было даже заказано специальное (и довольно дорогое) исследование рынка инструментов программирования.

И исследования рынка, и профессиональная работа по позиционированию продукта, и рекламная кампания были, разумеется, необходимы. На мой взгляд, все это надо было начинать гораздо раньше, вскоре после старта проекта, когда мы уже создали определенный «шум» в СМИ (интервью, статьи, дискуссии), и наш проект получил пусть небольшую, но все-таки известность. Но кто и как должен был это делать? В конце концов, мы-то были всего лишь разработчиками, поэтому организация и проведение подобного рода работ были прерогативой нашего инвестора.

Приходил очередной «маркетолог», выслушивал наши рассказы о принципах, целях и особенностях проекта и... уходил навсегда, после чего все повторялось с участием нового человека. Какие выводы следовало извлечь из проведенного исследования рынка, я так и не узнал. Кто наши потенциальные потребители и что надо было изменить в проекте, чтобы удовлетворить их потребности — так нам никто и не сказал...

Наш инвестор учился в Стэнфорде и Кингстоне и очень неплохо разбирается в бизнес-вопросах. Но — в бизнесе *вообще*. Он отлично знает, как организовывать дело, как поступать в тех или иных ситуациях. Он знает много «кейсов», которые разбирал, учась за рубежом. У него бывают свежие и нетривиальные бизнес-идеи. Но — идеи *вообще*. Он весьма умелый переговорщик — умный, по-хорошему хитрый, когда надо гибкий, иногда жесткий. Но вот специфику именно нашего проекта он почувствовал и осознал далеко не сразу. Вначале, наверное, ему казалось, что так как бизнес везде и всюду одинаковый, независимо от существа продукта, то и действовать следует так, как его учили в Америке... Выбор правильного позиционирования, поиск потенциальных клиентов и партнеров для такого сложного и не вполне обычного проекта, как наш,— все это произошло значительно позже. Тогда же действия носили характер спонтанных бросков без определенной стратегии.

И все-таки понемногу ко всем начало приходить понимание того, в каких сферах следует искать потенциальных заказчиков. Основными кандидатами виделись исследовательские центры, связанные с ВПК. С ними начали возникать какие-то контакты, переговоры. Инвестор организовал наше участие в конференции

в Ярославле в какой-то военной академии, и там мы представили наш проект. Конечно, само по себе выступление мало что могло дать — меня слушали в основном курсанты, которых явно пригнали только для заполнения зала. Гораздо важнее были контакты среди устроителей этого мероприятия и с другими выступавшими: именно они могли заинтересоваться нашими разработками.

Вершиной активности той поры был представительный семинар в одном московском оборонном НИИ ранней весной 1999 г. Семинар был целиком посвящен нам (если говорить откровенно, весь семинар был организован на деньги инвестора, и его тематику определяли именно мы), и в нескольких выступлениях мы смогли подробно рассказать обо всех аспектах нашего проекта — от концептуальных и технических до сугубо коммерческих.

Наши выступления получились весьма впечатляющими — сужу как по собственным ощущениям, так и по отзывам слушателей на последующем фуршете. Наверное, именно с этого семинара и следует начать отсчет «второй жизни» нашего проекта и самой нашей фирмы. Процесс начался; шел он очень медленно, каждый шаг давался большими усилиями, но, как показала последующая история, направление было выбрано правильное.

Но все это происходило уже без нас: к тому времени у ключевых участников возникли другие планы...

Исход и «вторая жизнь»

Саша Кротов закончил аспирантуру МГУ и в 1999 г. уехал в Финляндию: ни в Москве, ни в родной Перми никакие перспективы ему не светили, а в Хельсинки его ждали однокурсники с мехмата и фирма, в которой они уже работали. А в апреле 2000-го уехал и я: в цюрихском политехе меня ждал профессор Юрг Гуткнехт и позиция «обер-ассистента», что приблизительно соответствует нашему доценту: с обязаловкой в виде чтения лекций, но без профессорских привилегий.

Гендиректор покинул фирму практически одновременно со мной. С тех пор я с ним не встречался. Знаю только, что он оставил сферу ИТ, переехал в другой город и, кажется, занимается сейчас чуть ли не музыкальным бизнесом.

Автору виртуальной машины, по сути, никуда уходить не пришлось, он просто продолжил работу в своей прежней фирме, которую так и не стал оставлять ради перехода к нам. Еще раньше пришлось расстаться с двумя из трех молодых сотрудников: из-за неясных перспектив проекта и в связи с дефолтом 1998 г. инвестор сократил финансирование.

Для фирмы (остатков фирмы) настали трудные времена. Инвестор не скрывал своего разочарования и недовольства и выдвигал разнообразные «странные» идеи, вплоть до полной переориентации компании на совершенно другие сферы деятельности — типа выпуска мобильных телефонов...

Какое-то время фирмой руководил единственный из основателей, оставшийся в компании,— товарищ (бывшего) гендиректора. Надо отдать ему должное, он пытался твердо отстаивать «родной» компиляторный профиль компании от поползновений инвестора. Стратегия компании изменилась, о «коробочном продукте» никто уже не вспоминал, и основной упор новый гендиректор сделал на активный поиск заказчиков в среде изготовителей спецпроцессоров. Было довольно много переговоров, ребята встречались с представителями наших фирм и институтов, связанных с ВПК, выходили на европейские компании; эпизодически и я принимал «дистанционное» участие в этих делах. В результате всех усилий что-то, казалось, начало получаться. Все детали тогдашних перипетий мне было трудно отслеживать, бывая в Москве только эпизодически, однако тенденция, казалось, была позитивной.

Наша последняя встреча произвела на меня очень хорошее впечатление. Появились свежие идеи, были подписаны первые контракты. В фирме появились молодые ребята, некоторых из которых я знал еще по МГУ. Отношения с инвестором, еще год назад напряженные, постепенно переходили в более формальное и спокойное русло. Работа по совершенствованию компилятора продолжалась, ребята даже приняли участие в очередном совещании рабочей группы по стандартизации Си++ в Копенгагене.

Жизнь фирмы, казалось, налаживалась. И бог знает, как сложились бы дела, если бы у нового гендиректора все пошло бы так, как он задумывал... Но, к сожалению, не все в жизни зависит от усилий отдельных людей. Иногда в процесс вмешивается совершенно неожиданный случай, радикально меняющий ситуацию.

В небольшой статье невозможно рассказать все подробности последующей истории. Скажу только, что в один отнюдь не прекрасный момент все изменилось. Все предшествующие усилия если и не пошли прахом, то оказались под серьезной угрозой. Случился долгий период неустройства, кадровой чехарды, почти полного прекращения работ, поиска новых людей (иногда с тяжелыми ошибками в выборе кандидатов) и мучительного восстановления фирмы. Повлиять на этот процесс, находясь далеко от Москвы, я мог только отчасти...

Сейчас все перипетии далеко позади. Компилятор вполне жив, на его основе реализованы системы программирования для нескольких моделей российских микропроцессоров. Так что, вроде бы, все неплохо.

Но все эти разработки — традиционные полные компиляторы для конкретных машинных архитектур. Ни продвинутые языко-ориентированные инструменты, ни виртуальная машина С++ никому не нужны. Нетривиальные, явно инновационные идеи, когда-то заложенные нами в проект, не востребованы, никому из заказчиков не интересны, а ресурсов на самостоятельное создание и распространение законченных решений на основе этих идей нет.

И никто уже не вспоминает тот тяжелый и драматический, но захватывающе интересный период, который я попытался хотя бы пунктирно показать. Да и вспоминать-то некому: среди разработчиков не осталось *ни одного человека* из прежней команды... PC