Еще раз о программировании встроенных систем

В предыдущей статье автора, опубликованной в «КиТ» № 8'2006, были рассмотрены некоторые проблемы, возникающие при разработке встроенного ПО. Статья вызвала большой интерес читателей, и автор получил ряд откликов. В своей новой публикации автор отвечает на одно из писем.

Владимир ТАТАРЧЕВСКИЙ
arktur04@mail.ru

Очень хорошее письмо пришло от Русановского А. П., ведущего программиста-системщика. Приведу его в некотором сокращении со своими комментариями. Еще раз хочу подчеркнуть, что все нижеизложенное является исключительно моим личным мнением.

Здравствуйте, В. Татарчевский.

Вам пишет программист-системщик (29 лет отроду, со стажем работы порядка 7 лет, сейчас на должности главного конструктора, но делаю в фирме, как Вы пишете, все — от ТЗ до корпуса). Меня сильно задела Ваша статья в «КиТ» «Некоторые мысли по поводу программирования встроенных систем». Хочу поделиться своим мнением — по пунктам Вашей статьи.

1. *«Дурная спираль»: полностью согласен с описанием, но я это называю по-другому — «процесс разработки». Я считаю нормальным достижение между разработчиком и заказчиком компромисса. Да, это стоит времени, но устройство — это «лицо» фирмы, и заказчик должен быть доволен работой, несмотря на (чаще всего) «корявое» описание задачи в ТЗ. Чаще всего заказчик идет на дополнительное выделение времени на доработки и «непонятки». Правильно понять и разобраться в проекте — это наша профессиональная обязанность.*

Разумеется, разобраться в проекте — это наша обязанность, и, разумеется, дополнительное время на доработки нужно всегда. Но хотелось бы заметить, что согласование проекта программы с заказчиком до начала разработки позволяет значительно сократить объем доработок. Бывают и совсем «веселые» ситуации, когда заказчик не может четко сформулировать порядок работы прибора. Не разрабатывать же путем проб и ошибок!

Приведу пример из своей практики. Заказчик хотел получить устройство управления неким технологическим процессом (каким — не суть важно). Никакого ТЗ у него не было, имелось лишь руководство пользователя на аналогичный прибор немецкого производства, в котором содержалось весьма скудное и туманное описание работы устройства. Заказчику прибор нужен был срочно. Пришлось писать программу на основе имеющейся информации. Написали, проверили, вроде бы все работает, но как-то не так, не как в немецком аппарате. К тому же выяснилось, что программа не отслеживает ряд аварийных ситуаций техпроцесса, про возможность возникновения которых, с одной стороны, можно было догадаться, а с другой — они не были описаны нигде в явном виде. После нескольких попыток доработать программу «шаманским бубном» было принято кардинальное решение. Я взял бумагу, линейку и карандаш и за две недели нарисовал UML-диаграмму на 40 листах А4, после чего засел с инженером фирмы-заказчика (назовем его «технолог») за ее «отладку». И произошло небольшое чудо: все слабые места программы стали видны как на ладони. В результате технолог, ни разу в жизни с программированием не сталкивавшийся, не только объяснил мне те стороны технологического процесса, которые очень сложно было понять из первоначального «ТЗ», но и сам увидел в алгоритме работы системы участки, работу которых можно было существенно оптимизировать. В результате программа была полностью переписана за неделю и заработала практически сразу, причем алгоритм ее работы был существенно улучшен по сравнению с немецким прототипом. Трудно представить себе, во что вылился бы процесс отладки на реальном технологическом оборудовании.

2. *Для чего нужны «суперпрограммисты»? Ответ простой — вытаскивать фирму из, извините, д..ма, в которое она часто вляпывается по вине других сотрудников. Когда пять «программеров» писали, писали, а в итоге кроме исходников и через «бубен шамана» работающей программы нет ничего. Я вытаскивал три таких фирмы. Причем не скажешь, что эти люди не умеют разрабатывать, просто они делают это долго по сравнению с «суперпрограммистами». А теперь скажите, почему люди, которые работают быстрее и надежнее, не должны получать «суперзарплату»? Коммунизм кончился. В любой фирме есть проекты, которые нужны «вчера» и за которые платят чаще всего хорошие деньги. Вот такую работу выполнять должны «суперпрограммисты». Я всю свою профессиональную деятельность я занимаюсь только тем, что делаю «вчерашние» устройства. Именно такие люди вносят что-то новое, более гибкое (я первый внедрил в фирме автоматы с доработанной SWITCH, перевел фирму на ARM-процессоры, сейчас внедряю мультизадачку собственной разработки с кучей уже написанных драйверов, при этом выполняя все свои обязанности). Все это только ускорило разработку программ другими программистами в фирме. Наша фирма признана («Аргуссофтом») самой инновационной фирмой в России — мы больше всех внедряем новые технологии (как отечественные, так и зарубежные) в свои разработки, закладывая в них большой процент прочности на рынке, удобство использования. Я вас уверяю, что это делают не рядовые программисты-системщики. И за это я хочу хорошую зарплату. Если я пишу и думаю быстрее, почему это не использовать на заработок как мой, так и фирмы? И если я уволюсь — это не значит, что после меня никто не разберется в моей работе — я профессионал, вся документация оформляется честно (за это мне платят), дела всегда передаются, и я отвечаю на все вопросы даже после увольнения. Ну а если преемник не может разобраться — я же обучать не обязан, как говорится, ищите «спеца» (хотя чаще всего «спецы» просят такую же зарплату, потому что она выстрадана, и фирма только проигрывает от смены специалиста — проверенный факт).*

Вот оно, самое наболевшее. Зарплата и проекты, которые нужно было сдать «вчера». И суперпрограммист, который спасет всех, прямо как супермен в американском фильме. Но давайте разберемся по порядку. Что приводит к срыву сроков проектирования? Низкая квалификация разработчиков? Как правило, нет. Пишут медленно? Может быть. Но как показывает практика, основное вре-

мя уходит не на написание кода, а на его отладку, а также на разнообразные доработки, не предусмотренные в исходном ТЗ. Таким образом, проблема не в программистах, проблема в организации их работы. Не музыканты плохи, а дирижера нет, и ноты никто не раздал. И «человек-оркестр» (то есть «суперпрограммист») может только раз за разом спасать ситуацию, но изменить ее в корне он не может. Для этого ему нужно самому стать дирижером. Вот в этом и заключается мое принципиальное несогласие с автором письма. Он считает, что «суперпрограммист» должен работать за всех, вытаскивать фирму из кризисных ситуаций, а я считаю, что «суперпрограммист» должен руководить другими программистами и не допускать возникновения авралов. И, разумеется, вносить в работу фирмы инновации. Что же касается зарплаты, здесь, по-моему, двух мнений быть не может: чем выше квалификация специалиста и чем больший пост в фирме он занимает (то есть чем большую ответственность он несет), тем больше он должен получать. Обратное я никогда не утверждал.

Вы пишете о том, что оформляете всю документацию, и будете отвечать на вопросы по Вашим проектам даже после увольнения. И это очень хорошо, что Вы честно и профессионально относитесь к своим обязанностям. Не все программисты такие, к сожалению. И как тут не привести слова А. А. Шалыто: «Каждый, кто участвовал в крупном проекте по реконструкции ПО, навсегда запомнит то чувство беспомощности и потерянности, которое испытываешь, когда впервые видишь гору плохо документированных (но не всегда плохо написанных) исходных текстов. Доступность исходных текстов не слишком помогает, когда отсутствует доступ к ключевым разработчикам. Если программа написана на сравнительно низкоуровневом языке типа Си и плохо документирована, то все основные проектные решения обычно растворяются в деталях кодирования и требуют реконструкции. В таких ситуациях ценность документации более высокого уровня, такой как спецификация интерфейсов и описание архитектуры, может превышать ценность самого исходного текста» [2].

3. *Время и деньги. Чем быстрее программист выпускает программу, тем больше у нее шанс попасть на рынок и заработать деньги. Тут, я считаю, дело в поставленных технологиях разработки в фирме. Чем больше база навыков и специалистов у фирмы, тем быстрее ведется разработка. Кроме задач НИР, все остальные задачи прогнозируются по времени достаточно четко, чтобы сроки и деньги согласовать с заказчиком (конечно не с точностью до дня, но до квартала точно, а то и с точностью до месяца).*

Абсолютно согласен — дело в поставленных технологиях разработки.

4. *Одна фирма — один инструмент. Очень часто инструмент бывает с ошибками или есть новый инструмент, с помощью которого разработка ведется еще быстрее. По-моему, тут не все так просто. Все зависит от масштаба проекта и его стоимости. Хотя по большей части с этим пунктом согласен, но лучше его немного переформулировать: «один проект — один набор инструментов».*

Здесь тоже возразить нечего. Выбор инструментальных средств в немалой степени зависит от специфики проектов. Просто большинство небольших фирм специализируются в какой-то одной области, и проекты у них достаточно однотипные. Именно такого рода фирмы я и имел в виду.

5. *Программы нужно проектировать, а не писать. Совершенно верно, но только тогда, когда проект нужен не «вчера». Иначе пока вы будете проектировать, другие будут продавать. Большие проекты с грифом «вчера», можно выполнять одновременно с проектированием. Обычно большая часть программистов прекрасно понимает, что от них требуется, даже пока нет проекта. А лучше, конечно, иметь наработки (которые обычно предлагают «суперпрограммисты»). Заказчик «подсаживается» даже на программу с ошибками, но чаще всего на ту, которая быстро выпущена — в ней ошибки со временем исправятся, ну а деньги заказчика останутся в нашей фирме. Скорость, скорость и еще раз скорость, ведь время — деньги (конечно не в ущерб качеству).*

Ключевая фраза здесь: «пока вы будете проектировать, другие будут продавать». Именно в этой фразе отражен менталитет большинства разработчиков и их руководителей. Интересно, почему никому не приходит в голову построить без проекта, к примеру, ракету? Чего там сложного? Возьмем напильник и за выходные в гараже сделаем. Что время терять на какие-то там чертежи? Скорость, скорость и еще раз скорость, вам не за бумагу деньги платят, а за ракету, а если что не так, мы потом напильником доработаем. А документацию потом напишем, когда ракету запустим.

А может быть, лучше наоборот? Пока вы будете проектировать, другие будут писать код. Пока вы будете писать код, другие будут свой код отлаживать. Когда вы начнете продажи, другие будут по-прежнему отлаживать код. Пословица «поспешишь — людей насмешишь», актуальна в программировании точно так же, как и везде. «Впарить» заказчику сырое изделие, а потом устранять ошибки по ходу дела — плохая бизнес-стратегия, как мне кажется, хоть она и встречается на каждом шагу. К тому же за устранение ошибок заказчик платить не обязан, а вот ваша фирма за работу своих программистов платит. Как бы в убытке не оказаться.

Спасибо что дочитали все это. Может, где-то эмоционально написал — простите. К сожалению, это реальная сторона наших разработок. Не забывайте: человек — это маленькая фирма со своими амбициями, которая тоже должна иметь право зарабатывать.

Совершенно верно. Человек — это маленькая фирма. И человек, и фирма проходят разные этапы в своей жизни. Как очень верно написал И. Каршенбойм: «… речь… идет не только о «железках», софте, но и о документации. То есть предприятие прошло этап «спаяем на коленке кое-как и спихнем». Если речь идет о промышленной разработке документации, значит, на этом предприятии работают люди, которые задумываются о том, как оно будет работать в дальнейшем» [3]. ■

Литература

1. Татарчевский В. Некоторые мысли по поводу программирования встроенных систем // Компоненты и технологии. 2006. № 8.
2. Шалыто А. Новая инициатива в программировании. Движение за открытую проектную документацию. **www.is.ifmo.ru**
3. Каршенбойм И. Здравствуйте, дорогие читатели! // Компоненты и технологии. 2006. № 8.