проблемы создания CRM-систем: взгляд изнутри

крис касперски, no-email

если бы программисты строили дома, то первый залетевший дятел разрушил бы всю цивилизацию. у этой проблемы есть два решения: строить дома по уму или создать автоматизированную систему отстрела дятлов, которая в свою очередь потребует еще одной системы для своего обслуживания и так до бесконечности как в старой арабской сказке. качество программного обеспечения невелико и продолжает ухудшаться с каждым годом. почему? автор, принимавший опосредственное участие в создании нескольких CRM систем, решил поделиться с читателями некоторыми пикантными особенностями их производства

неприятно, когда тебя не понимают. Ещё более неприятно, когда понимают неправильно

народное

введение

Пока моя мать не пошла работать в СЭС она была нормальным человеком, пока мой друг не устроился на мясокомбинат он ел колбасу, пока моя (бывшая) жена не попала на консервный завод она любила томат… Это только снаружи все выглядит таким красивым, вкусным, блестящим, питательным и… безопасным. Но стоит только узнать из чего делают колбасу… Ой, нет, не надо про колбасу.

Кстати говоря, для реконструкции ее состава совершенно необязательно устаиваться рабочим на завод или заниматься промышленным шпионажем. Возьмем пельмени (почему бы и нет?!), посмотрим на цену и попытаемся скалькулировать из какого сырья их производят, чтобы выйти хотя бы на нулевой уровень рентабельности (про то, что некоторые уроды используют бетономешалки, в которые вместе в фаршем попадают крысы и другие незапланированные составляющие мы скромно промолчим).

Производить пельмени из качественного сырья невозможно потому, что стоить они будут _намного_ дороже, чем у конкурентов, а потребитель в первую очередь все-таки смотрит на цену, особенно в тех случаях, когда критерии качества настолько расплывчаты, что практически не поддаются оценке.

Какое это имеет отношение к программному обеспечению?! Самое прямое! Качественных программ нет и не будет! Повсеместно используется дешевая рабочая сила (неквалифицированные программисты) и некачественное сырье (системы быстрого проектирования). Писать программы "как надо", во-первых, слишком дорого, а, во-вторых, никто в действительности не знает как управлять действительно большими проектами (к которым относятся CRM-системы).

Потребители верят, что в крупных компаниях дураков не держат, а если и держат, то на вторых ролях. Ведь не идиоты же там работают! Качество программной продукции, конечно, вызывает определенные нарекания, но никаких поводов для паники у нас нет, ведь так?!

На самом деле, процесс создания больших систем просто ужасен и это просто чудо, что программы вообще запускаются и не падают от малейшего дуновения ионного ветерка. Ошибок (известных) там столько, что… Но если об ошибках знают, почему их не исправляют?! Вопрос носит философский характер, однако, автор все же попытается ответить на него.

системы под ключ, но ключ не под замок

Customer Relationship Management Systems (Системы Управления Взаимодействием с Клиентами или, сокращенно, CRM[S]) — это сложные, многокомпонентные механизмы, включающие себя и базы данных, и модули обработки информации, и еще много чего. И все это хозяйство должно быть синхронизовано со многими каналами связи (от традиционных call-центров, до мобильного сервиса) и органично интегрировано в уже существующие корпоративные системы, в противном случае, вместо ожидаемого увеличения продаж мы получим только огромную головную боль с необходимостью радикальной реорганизации уже

сложившегося уклада. А это значит, что эффективность сотрудников от самого нижнего до самого верхнего звена неизбежно снизится на некоторое время. Увеличится количество ошибок, совершаемых операторами, многие из которых если не фатальны, то весьма болезненны.

CRM-системы общего назначения, созданные компаниями-гигантами (типа Microsoft Dynamics CRM), достаточно хорошо отлажены, однако, требуют двухсторонней адоптации для своего внедрения. Вы тянете Dynamics к своему бизнесу используя лишь часть его функций или адоптируя его функционал под конкретные нужды, но… отечественных реалий зарубежный функционал не понимает и парни из отдела поддержки рекомендуют адоптировать вам свой бизнес, притянув к Dynamics'у. Это только на бумаге все выглядит красиво и гладко, а в действительности…

Четко налаженный бизнес, уже отработанный годами, не может радикально улучить свою рентабельность никакой автоматизацией. Да, компьютеры позволяют разгрузить сотрудников (уменьшив их число), развернуть системы он-лайновых и мобильных продаж, точнее прогнозировать объемы закупаемого товара, но… автоматизация таит в себе _огромную_ угрозу делая бизнес небезопасным.

Возьмем, например, штрих-код. Изобретение, бесспорно, удобное и вроде бы надежное, но! Злоумышленники уже давно развлекаются тем, что печатают свои собственные этикетки со штрих-кодами, незаметно наклеивая их на товар, чтобы снизить его цену в несколько раз. Конечно, мошенников ловят. И даже сажают. Но… магазины продолжают нести убытки. Утомленный кассир часто не замечает, что подносит к сканеру банку с черной икрой, а на терминале вспыхивает "спички, одна коробка".

Автоматизация продаж и централизованное хранение информации о товарах, сделках, клиентах, открывает огромный простор для всевозможных хакерских атак и такие атаки совершаются с завидной регулярностью. Древние говорили по этому поводу: не клади все яйца в одну корзину. Думаете, они заблуждались? Так почему же вы в восторге от того, что все данные стекаются к центральному серверу, зачастую наделенному полномочиями принятия решений, касающихся новых закупок/отгрузок и хранящего конфиденциальные данные клиентов?!

Вот три основные угрозы: 1) разрушение (или искажение) данных, влекущее за собой "паралич" торговли со всеми вытекающими убытками; 2) хищение конфиденциальных данных, последствия которых очевидны; 3) принуждение сервера к совершению "левых" переводов денег и/или товаров.

Таким образом, чем "плотнее" CRM-система интегрирована в систему корпоративной торговли, тем выше требования к ее надежности, иначе убытки от атак рискуют пересечь тот рубеж, что вгоняет компанию в штопор неизбежного банкротства. Слабая интеграция CRM-системы с одной стороны сводит возможный ущерб от атак к минимуму, но и не дает никакой полезной отдачи. Фактически, CRM-система при этом превращается в пятое колесо. Интегрировать его в бизнес мы интегрировали, но пускать его в ход из соображений безопасности не собираемся.

Многочисленные мелкие фирмочки предлагают создание CRM-систем "под ключ", в результате чего их интеграция протекает без отрыва сотрудников от производства и без перестройки чего бы то ни было. Весьма соблазнительные перспективны. Не каждый менеджер устоит перед искушением.

Но… никакая фирмочка не в состоянии создать настоящую CRM-систему. Это все равно, что трое студентов спроектируют и запустят в космос спутник вместе с ракетой-носителем. Сложность и многокомпонентность CRM-систем требует привлечения специалистов из самых разных областей, а так же огромное количество управленческого персонала, поскольку на управлении производством замыкается абсолютно все, а без него сотрудники превращаются в реальное стадо. Стоит только дать программистам свободу и вместо Капитолия мы получим груду камней. А кормить аппарат управления можно только за счет массовых продаж, иначе бизнес станет нерентабельным и коллапсирует в черную дыру, еще до того, как продукт будет отлажен хотя бы наполовину.

Причем эта половина будет осуществляться… как бы это поделикатнее сказать… в процессе интеграции CRM в ваш бизнес и все ошибки программистов станут вашими прямыми убытками. Хотите ли вы финансировать отладку CRM за свой счет (изначально не включенный в договор) и увеличивающий реальную стоимость интеграции CRM на несколько порядков?!

Намного выгоднее приобрести CRM общего назначения от крупного производителя (естественно, не первой версии, которая была сырой даже у Microsoft) и смириться с неизбежными издержками адоптации своего бизнеса под чужеродный ему продукт. Во всяком

случае, этот продукт уже отлажен (вы же верите в это, правда?) да и за безопасностью Microsoft следит намного лучше, чем группа кустарей.

А вот тут вы сильно не правы и позвольте мне вас поправить. Кустарно написанную CRM от неизвестно кого, атаковать… ну не то, чтобы совсем невозможно, но весьма затруднительно. Да и неинтересно, поскольку она существует в единичном экземпляре, недоступным для непосредственного изучения до тех пор, пока его не удастся утянуть, а утянуть его обычно бывает непросто.

Массовую CRM-систему "тянуть" не надо. Ее даже не требуется покупать. Пока существует Интернет, будет существовать и варез, а, значит, получить точную копию массовой CRM-системы не составит большого труда. А при удачном стечении обстоятельств и добыть исходные коды. Закрытые конечно, но… чем больше компания, чем чаще происходят утечки.

Все знают, что исходный код Windows 2000 был украден у компании VM Ware, выпускающей одноименные виртуальные машины. На самом деле, эта кража не единственная, просто самая громкая и скандальная, а украденный комплект исходных текстов очень сильно неполный (в архиве ~200 метров). Помимо этого было как минимум две других кражи. Архивы Windows 2000, весящие свыше 700 Метров, несут практически полный исходный код ядра и некоторых прикладных программ. Собрать из него рабочую Windows, впрочем, еще никому не удалось, но зато упростило поиск ошибок в системе.

Кража исходных кодов XP и Server 2003 прошла незаметно и вот совершенно недавно один из бывших сотрудников Microsoft перед увольнением раздавал со своего приватного ftp-сервера _полные_ исходные коды Server'a 2008 RC2 всем желающим. Ну, может быть, и не полные… Я не качал, не знаю. Просто взглянул на размер архива — 7,4 Гбайт плюс еще один архив с инструментами для компиляции и сборки всего этого хозяйства. Размер внушает уважение! Через несколько дней ftp-сервер прекратил свое существование, но исходные тексты уже просочились в файлообменные сети, так что пускай теперь их ФБР оттуда выковыривает. Если, конечно, сможет, в чем я очень сильно сомневаюсь.

Надеюсь, читатель простит мне это небольшое отклонение от темы, которым я хотел показать, что "безопасность" это всего лишь иллюзия и жалкий самообман. Безопасны лишь продукты аля "Неуловимый Джо" который неуловим не потому, что крут, а потому что никому не нужен!!!

у CRM под капотом

История — одна компания (имя которой я не скажу, потому что вы ее все равно не знаете) внезапно озаботилась тем, что у конкурентов уже есть CRM, а у нее нет. Поскольку, это была довольно крупная компания с развитой сетью магазинов по всему СНГ, то она не стала размениваться по мелочам, а наняла консультанта по CRM-системам, который опасаясь быть битым за выбор неудачного продукта (ну да, попробуй порекомендовать Dynamics — за его глюки реально побить могут), он предложил: а давайте объявим тендер (тендеры — это ведь так модно, современно и актуально!) на разработку CRM среди отечественных софтверных компаний (почему отечественных? а они лучше знают реалии нашего рынка и вообще, свои — не чужие).

Тендер выиграла компания имя которой вы наверняка знаете, но я его все равно не скажу, поскольку мне с ней еще работать, а руководство не шибко любят, когда не только выносят сор из избы, но еще и говорят, что это за изба такая и кому она принадлежит. В конечном счете, это не так уж и важно. Мы же ведь не желтая пресса, правда? Так что обойдемся без имен.

Пикантность ситуации в том, что кроме громкого имени у компании за душой не осталось ничего. Все умные сотрудники давно свалили за рубеж или к конкурентам, а потому программировать было реально некому. А программировать надо, потому как договор уже подписан и сроки обозначены. Вот только денег нет. Короче, ситуация… Зачем было подписывать договор, если нет ни денег, ни программистов?! Странный вопрос… А вы бы упустили такой шанс?! Да весь наш бизнес действует по принципу: сначала мы ложимся под договор, а только потом начинаем думать под чем мы подписались и во что ввязались.

Информация к размышлению. Китайцев много. Очень много. И все они ужасно голодные. И не потому что, вместо вилок у них палочки, а потому жрать нечего. Оутсортинг весьма модное слово, но мне больше нравится термин "субподрядчик". Главное, что все законно. Проблема однако же в том, что китайцы (в своей массе) проектированию не учились и программируют не так, как надо (с учетом всех рисков и просчета всевозможных ситуаций), а так, как колесо Сансры на ~~душу~~ ум положит. Простые программы получается довольно

неплохо, но толпа каменщиков, среди которых не значиться ни одного архитектора, башню не построит и в какой-то момент сооружение начнет разваливаться на части (особенно, если это крупное сооружение, а не шалаш). Короче, когда код начал выходить из-под контроля и проект потерял управляемость, а ошибки стали плодиться как австралийские кролики в брачный период, множась в геометрической прогрессии, китайцы внезапно растворились. Найти куда ведут концы так и не удалось. Проект был готов на 90%, но _никак_ не работал, потому _все_ его компоненты были готовы на 90%, и ни один на 100%, причем код был написан так, что смотреть на него и не хвататься за валидол — это же какую выдержку надо было иметь!!!

Ну нету китайцев — и не надо! Главное, что есть оставленный ими код, который теперь можно… э…. нет, "переписать с нуля" не подходит! Это вообще не вариант. Да заказчика инфаркт хватит, а инфаркт — это негуманно. Руководство приняло мудрое решение, а точнее даже целых два. Продолжать _сопровождать_ код (еще один политически корректный термин, позволяющий говорить одно, а иметь ввиду совершенно другое), предварительного "заморозив" его, то есть запретив вносить какие бы то не было изменения в уже написанные строки. Дурдом?! Вовсе нет! Основной принцип программирования. Можно даже сказать фундаментальный — "Не трожь, а то упадет".

Сопровождать код было поручено другой фирме. На этот раз российской. Которая на самом деле оказалась американской (по месту регистрации), только среди ее сотрудников ни одно американца так и не обозначилось. Американца — это в смысле по паспорту. Зато была~~о~~ куча индусов. А у индусов всегда много хорошей травы. А трава — это креатив. Без шуток!!! Недостающие 10% функционала были написаны за рекордно короткое время. И проект ~~(точнее его ядро)~~ удалось собрать из разрозненных модулей воедино. Компиляторы выдавали тысячи предупреждающих сообщений, но… все-таки соглашались ~~е~~генерировать исполняемые файлы, которые даже кажется запускались… И при удачном стечении обстоятельств падали не после первого нажатия на клавишу, а после второго, что приводило индусов в восторг, совершенно не разделяемый заказчиком, который к тому времени ~~был~~ ходил мрачнее тучи.

Ах да… совсем забыл сказать. Это было только ядро. А к нему еще требовалось прикрутить пользовательский интерфейс. Отвлекать индусов от борьбы с багами такой ерундой не решились и наняли бригаду голодных студентов которые в перерывах между сексом и сессией (что по сути одно и тоже) "лабали" интерфейс. Причем одна часть студентов лабала его на DELPHI, а другая — на .NET. Для тех кто не знает — .NET это VisualBasic, замаскированный под Си и перенявший худшие черты от Java, Си и, собственно, самого Бейсика. Тихий ужас… Но нет, ужас нас ждет впереди.

Короче, мы имеем продукт, готовый к демонстрации перед заказчиком, с замороженным "китайским" ядром (примерно полмиллиона строк) и кучей "надстроек", написанных индусами, реализовавших недостающие 10% функционала всего в двух миллионах~~ов~~ строках. Грубо говоря, индусы достроивших~~ли~~ здание~~,~~ путем помещения его в саркофаг, накрытый сооружением из фанеры/бетона/картона и обклеенного обоями с узором в стиле "юный дизайнер впервые увидел Photoshop".

На заказчика продукт произвел весьма благоприятное впечатление (обои скрывали весь хаос, царивший под ними) и проект перешел в стадию ранее именуемой "пуско-наладкой". То есть, доводкой изделия до ума, но уже на рабочей площади заказчика.

низший пилотаж

Продукт исправно проходил все тесты, но… не работал. Ну, не то, чтобы совсем не работал. Но работал явно не так, как это задумывалось и уж точно не так, как этого ожидал заказчик. Одно падение следовало за другим, память утекала стремительным домкратом, занесенных в базу данных~~х~~е там не оказывалось, ну а потребности в ресурсах превзошли возможности даже самых мощных компьютеров из тех, что имелись на рынке серверов.

Исполнитель во всем винил Microsoft и ее глючную ось, отчаянно ругал Intel и предлагал все переписать под Sun Solaris, смутно представляя себе, кто будет это кодить и какого качества окажется полученный код. Заказчик воспринял такое предложение без должного энтузиазма и предложил выкинуть 90% функционала, ценою доведения до ума оставшихся 10%. Исполнитель сначала отчаянно завыпендривался, но потом понял, что другого выхода из ситуации все равно нет и принялся подключать к проекту новых специалистов~~, в число которых попал и автор этой статьи.~~

К тому времени, CRM-система находилось в минимально рабочем состоянии и даже была внедрена в нескольких мелких филиалах компании заказчика. Основную проблему представляли внезапные отказы, происходящие в самый неожиданный момент, и практически

ни один из них так и не удалось воспроизвести, то есть выделить последовательность действий, приводящую к краху. Крахи происходили совершенно стихийно, не подчинясь ни логике, ни смыслу…

Бригада "скорой помощи" (в которой и оказался автор этой статьи), возглавляемая мудрым программистом, на ходу решающим проблемы, о существовании которых автор не то что не подозревал, но даже никогда не задумывался, с самого начала попыталась поставить вопрос ребром. Мы имели богатый опыт работы в таких ситуациях, мы знали, чем вызваны крахи и самое главное, у нас был четкий план действий, которого не было у других, даже у Гайдка и то не было такого плана!

Наши требования: 1) "разморозить" китайский код, позволив нам вносить в него изменения; 2) прекратить наклеивать обои, пока не решены проблемы с фундаментом; 3) хватит сначала программировать, а потом думать, давайте же, наконец, сначала формулировать задачу, потом искать решение, и только убедившись в его верности тянуться к клавиатуре.

Увы! Наши требования не были выполнены и нам пришлось совершать эквилибристические трюки, где хватило бы и посадки на шпагат. Ладно, хватит абстракции, переходим к конкретике. В китайском коде обнаруживается оператор return, совершающий преждевременный выход из функции при котором определенные структуры данных оставались ются неинициализированными, то есть в них мо гложет содержаться все, что угодно. Отсюда и стихийность крахов. Это явная ошибка. Не наша. Китайская. Устранялась ется она тривиально — заменой оператора return на break. Но… китайский код заморожен и трогать его ни при каких обстоятельствах нельзя. Это приказ. К тому же, нам было сказано, что обозначенный модуль прошел все тесты и не споткнулся ни на одном из них. В общем, мы кругом не правы и страдаем какой-то ерундой, когда тут программа рушится.

Наше начальство почему-то думало, что глобальные крахи требуют глобальных причин и какой-то жалкий return, воткнутый вместо break'a, не может быть их причиной. Опять-таки тесты… Судя по тестам все ОК. Плевать на то, что ошибка видна невооруженным глазом. С модулем все ОК и править его нельзя. Я начальник — ты дурак. То есть, наоборот. Мы не были ни начальниками, да и дураков среди нас не водилось. Ладно… мы подписались под эту работу и мы доведем ее до конца.

Проанализировав условия, при которых срабатывает этот return, мы обернули китайскую функцию несколькими тысячами строк кода, гарантирующими, что эти условия никогда более не возникнут. Это стоило нам недели работы и мы до сих пор не уверены, что учли все возможные варианты и что наш код правильный. Но… чем больше мы работали над программой, тем реже она падала. К сожалению, работать приходилось по полной и извращаться так, как никто еще не извращался.

А вот другая китайская функция. Сначала освобождает память и тут же, словно спохватившись, считывает оттуда хранящиеся там данные. Естественно, после освобождения памяти, этот блок возвращается в пул свободных блоков и может быть выдан любой другой функции. Но вообще-то, в данной ситуации, на однопроцессорных машинах, вероятность того, что кто-то успеет запросить блок в промежутке между его освобождением и повторным использованием была близка к нулю, поскольку, на однопроцессорных машинах в каждый момент времени выполняется только один поток и шансы на то, что переключение потоков произойдет на заданном отрезке кода были действительно невелики. Но вот на многопроцессорных машинах (или процессорах с несколькими ядрами), где одновременно исполняются более одного потока, программа падастла не проработав даже и нескольких часов!!!

Меняем две строчки местами — решаем проблему! Увы, править замороженный код нам никто не дает. Что делаем мы? Обсудив несколько вариантомв, останавливаемся на наименьшем зле – пишем свой собственный аллокатор. То есть ту штуку, которая выделяет память. Нам потребовалось написать очень хитрый аллокатор, который гарантирует, что содержимое памяти останется неизменным даже после ее освобождения. Самое простое было бы выделять память в обратном порядке, при котором освобожденная память выделяется последней, но это привело бы к увеличению потребности в памяти (которой и так не хватало из-за прожорливого интерфейса, представляющего собой внебрачный гибрид DELPHI с .NET) и нам пришлось писать автоматический анализатор машинного кода, вылавливающий ошибки данного типа и "придерживающий" освобожденный блок памяти только в тех случаях, когда это действительно необходимо.

Частота падений программы резко сократилась, но все еще превышала тот уровень, с которым было можно работать и мы продолжили анализ кода дальше. Ага! Вот еще одна функция производит запись по неинициализированному указателю. Ошибки такого типа

называются "ударом по памяти" и очень коварны. Анализ показал, что данный указатель как правило принимает такие значения, что запись затрагивает неиспользуемые регионы памяти, не производя никаких разрушений, но при определенных обстоятельствах, под удар попадают ценные данные, после чего программа идет в разнос (а если не идет — тем хуже для программы!!! Как вам если сумма в 5 рублей незаметно превратиться в 634673672 или что-то типа того?!). Пикантность заключалось в том, что никаких путей инициализации данного указателя извне функции не существовало (или мы их просо не нашли? но теперь это неважно). Естественно, наше требование разморозить код было вновь отклонено с неизменной мотивацией: "судя по тестам, с этом модулем все ОК". Ну как, как, черт возьми, объяснить, что с _этим_ модулем _действительно_ все ОК, но он гробит посторонние модули, создавая иллюзию, что ошибка в них, когда она совсем в другом месте, и это место в нем!!! И вот оно — это место!!! Мы видим ошибку, мы обсчитываем все условия ее проявления. Устранить ее… ровно _два_ символа дописать. Так ведь нет… не дают!!!

Признаюсь, мы поступили нечестно. Мы нарушили запрет. Мы модифицировали замороженный код. Потому что… не видели других путей. Исходные тексты, впрочем, остались нетронутыми. Мы хакнули функцию прямо в памяти. Именно хакнули, потому что "хаками" называются кривые приемы программирования, к которым приходится прибегать, когда не срабатывают (или запрещены) стандартные. Нам нужно было модифицировать всего одну ячейку, адрес которой был неизвестен и менялся в зависимости от ключей компиляции, а потому "хакерство" обошлось довольно дорого в смысле затраченного времени. К счастью, у нас уже был анализатор кода, осталось лишь доработать его и… наша функция правила баг "на лету", но руководство об этом не знало (оно вообще не вдавалось в технические подробности, за что мы безумно ему благодарны, иначе бы нам ни за что не удалось довести нашу работу до конца, а мы таки ее довели).

Итог: среди полумиллиона строк китайского кода мы нашли и исправили свыше трех тысяч ошибок. Из них порядка двух тысяч обнаружили наши утилиты для автоматизированного поиска, а оставшуюся тысячу ~~мы~~ откопали мы сами (лично я нашел менее пятидесяти, что при численности группы в девять человек выглядит не слишком впечатляюще, так что не нужно думать, что я лучший, лучшим я никогда не был, но разве это повод для комплексов?! зато мне посчастливилось поработать бок-о-бок с действительно лучшими программистами).

в погоне за монстром

Забудьте все, что я вам сказал. Я ругал руководство? Когда?! Да, ну, ерунда какая… С политической точки зрения руководство приняло пускай не лучшее решение, но все же вполне логичное и обоснованное, оставаясь ему верным до самого конца. В противном случае… тот стиль, которым мы правили ошибки, работал лишь благодаря неизменности китайского кода. Если бы кто-то начал его переписывать, воцарился бы полный хаос и проект провалился бы в тар-тарары, а так… заказчик получил вполне работоспособную CRM-систему. Он работает с ней уже больше года и верит ей как самому себе. Но только мы знаем, что эта тварь может рухнуть в любой момент, поскольку там все держится на соплях и многочисленных подпорках. И никто (даже мы) не способен сказать, что произойдет, если отклеить обои: рухнут ли стены или нет.

Виноваты ли китайцы? Я уже молчу про индусов. Это вообще отдельная история, но поскольку их код не был заморожен, его удалось исправить малой кровью. Надеюсь, вы понимаете, что "китайцы" и "индусы" даны не в отвлеченном смысле, как вся нация в целом, а исключительно в контексте кода, написанного отдельными их представителями? ОК, если так, можно продолжать повествование.

Китайцы не виноваты. Они программировали как могли. Как их учили (и кто их учил?!) Ошибки в нашей — программистской — работе неизбежны. Конечно, чем квалифицированнее программист, тем меньше ошибок он совершает, но специалисты очень дороги — работать меньше чем за сумму с тремя нулями сзади и двумя цифрами спереди — это вообще оскорбление, а работать над "тупым" кодом это уже не оскорбление, а осквернение всего высокого и святого. То есть без дешевой рабочей силы нам никак не обойтись.

Но от каменщика (кодера) высокая квалификация и не требуется! А чтобы он клал кирпичи ровно и здание не накренилось как Пизантская башня, нужен учет и контроль. То есть, проблемы управления большими проектами сводятся к проблемам менеджмента. К сожалению, эта проблема неразрешима. Потому что, чтобы управлять, нужно иметь план. А план - это последовательность действий. То есть программа. Следовательно, управление проектом это и есть его программирование. И менеджер по сути такой же программист, как и все остальные.

Но, раз он программист, то он требует управления. Генерального менеджера. По сути дела еще одного программиста, только намного более абстрагированного от технических проблем и оперирующего не битами и байтами, а ресурсами совсем иного рода — финансами, людьми, etc.

Рыба гниет с головы, а в ошибках винят программистов. Не менеджеров. Знали бы потребители в каких условиях тем программистам приходится работать, ведя борьбу на два фронта — как против багов, так и против менеджеров… Но без менеджеров программисты вообще не работают, так что выбирать не приходится.

вместо заключения

Чем вызвана закрытость исходного кода большинства коммерческих программных продуктов? Секретностью? А что там скрывать?! Все алгоритмы защищены патентами, а кому нужно будет что-то украсть, тот просто возьмет готовый бинарный модуль или в крайнем случае реконструирует исходный текст с помощью дизассемблера. Отсутствие исходных текстов еще никого и ни от чего не останавливало и не остановит.

Компании отказываются раскрывать исходные тексты серьезных продуктов, потому что бояться, что окружающие увидят их тупость, оценив ее по достоинству. И это справедливо как для мелких фирм, так и для огромных корпораций.

Дмитрий Катков (издатель фото-журнала "Хулиганствующий Элемент" — http://www.photo-element.ru/) не боялся летать на самолетах, пока не начал разрабатывать программное обеспечение для них. Как говорят, существуют две категории лохов: первые советуют все подряд, вторые утверждают, что ничего нельзя сделать.

Возникает вопрос, к какой категории я отношусь, если осмеливаюсь утверждать, что для повышения надежности программного обеспечения (в том числе и критического) _ничего_ нельзя сделать? Возможно, я просто пессимист? Но ведь, как гласит другая поговорка, пессимист — это хорошо информированный оптимист.

Знаете, если бы я не участвовал в создании CRM-систем, то ни за что бы не писал статью в таком ключе. Но к черту очередной восторженный (или критический) обзор. Вы хотели знать как создаются CRM-системы? Можно ли им доверять? Теперь вы это знаете. А знаете ли вы, что иные знания даются ценой изгнания из рая?! Очень надеюсь, что меня после публикации этой статьи не прикуют к скале цепями как Прометея и я продолжу участвовать в разработке CRM и других систем, потому, что эта та работа, которая мне реально нравится, пусть даже я не лучший, но судьба в очередной раз предоставляет мне шанс поработать с лучшими!!! Система телеметрии нефтяной трубы (тихий ужас и просто кошмар — в смысле количества ошибок на квадратный байт). Но… это уже тема для следующей статьи.

>>> врезка квалифицированный специалист - это пупс, который, успешно преодолевая локальные сложности, упорно движется к глобальной катастрофе

Любая идея, доведенная до абсурда, вырождается в собственную противоположность. Именно такая ситуация складывается в североамериканской промышленности ПО с RAD-средствами разработки. В некоторый момент инструменты, предназначенные для быстрой разработки приложений, стали вытеснять в умах менеджеров все остальное, в том числе разработчиков, заказчиков и сам проект. Неоправданное, гипертрофированное внимание к Процессу в ущерб другим факторам разработки породило массу формальных процедур — но качество полученных продуктов и количество успешных проектов оказалось разочаровывающим.
(c) Арсений Чеботарев

>>> врезка позднее принятие решений

"Поздний анализ" означает "принимайте конкретные решения только тогда, когда это нужно". В большинстве случаев принятые на начальной стадии решения относительно кода приходиться отменять под влиянием новых требований или других обстоятельств. ***Не принимайте никаких решений по поводу кода, которого у вас еще нет, — тогда вы развяжете себе руки при реализации текущих задач.***
(c) Арсений Чеботарев

>>> маркетоиды превращают хороший продукт в дерьмо, а дерьмо — в товар!

>>> проблема развитого ума:
увидеть проблему и оправдаться в ее необходимости