ПРОГРАММИРОВАНИЕ*, АНАЛИЗ И ПРОЕКТИРОВАНИЕ СИСТЕМ*

Однажды программисты погубят этот мир

nckma 14 сентября 2016 в 19:39 67,2k

1. История первая: воспоминание

Когда я был маленьким и ходил еще в младшую школу, со мной приключилась такая история. На перемене я с несколькими друзьями находился в классе, когда с полки сам собой упал цветок. На нашу беду тут же вошла наша учительница и безо всяких разбирательств обвинила нас в хулиганстве. Красной ручкой были сделаны записи о плохом поведении в дневник, вызваны родители. Это было так обидно и не понятно, что навсегда врезалось мне в память. С тех пор я часто размышлял,

что же заставило цветок упасть?

А ведь самое логичное объяснение — он сам упал. Просто он рос, выбрасывал новые побеги, развивал и наращивал массу. Тянулся к солнцу. Однажды проекция центра тяжести цветка вышла за пределы опоры и он опрокинулся.

Не знаю, зачем я это все пишу. Возможно меланхолия пришла вместе с осенью и желтыми листьями. Иногда мне кажется, что наука программирования так стремительно развивается, так идет вперед семимильными шагами, так быстро создает новые технологии, что однажды человек не сможет все это обуздать.

2. История вторая: программисты не знают точно, как работают их программы

Я довольно хорошо помню свою первую программу за которую мне заплатили деньги. Было это очень давно. В те стародревние времена компьютеров как таковых у нас в стране еще не было. По крайней мере они были очень большой редкостью. Некая коммерческая организация, а тогда такие организации назывались «кооператив», затеяла собственное производство компьютеров.

Это был улучшенный клон компьютера «Специалист» на базе 580-го процессора. Предполагалось создать полностью свое ПО для этого компьютера. Первый этап — написание БИОС. Ну а я делал для БИОСа самую первую процедуру: тест ОЗУ. Пикантность ситуации была в том, что производимый компьютер был настолько не надежен, что для процедуры теста ОЗУ нельзя было использовать временные переменные в памяти, нельзя ничего хранить в этом самом ОЗУ пока оно не протестировано. Ну это в общем логично, конечно. Однако, как можно было написать тест памяти имея всего навсего восьми битный аккумулятор A и еще шесть восьми битных регистров общего назначения B, C, D, E, H и L? При этом тест должен на графическом экране в правом верхнем углу писать сколько памяти протестировано: 8 Кб, 16 Кб, 24 Кб… Делал я эту программу долго, наверное два или три дня. Написал, получилось кажется до сотни строк ассемблерного кода, проверил и, гордый своей работой, сохранил программу на магнитную ленту. Пришел сдавать-показывать работу, а магнитная лента не читается на магнитофоне. Ну что делать: тут же в офисе на рабочем «Специалисте» по памяти (!) весь код заново набрал, запустили, проверили — ура! работает! Получил кажется рублей 30 за работу. Большие деньги.

В те очень далекие времена я в своей программе понимал АБСОЛЮТНО ВСЕ.

С тех пор, так получилось, я писал программы на ассемблере процессоров 6502, Z80, x86. Потом заболел языком Forth. Кое что получалось на FoxPro. Потом настала пора C/C++. Говорили, кто

поймет MFC, тот вообще специалист высшего класса. Выучил-освоил MFC. Писал драйвера для Windows 2000/XP/w7 и для Linux. Нужно было делать сайт для компании — участвовал: html, css, js, php. Еще немного микроконтроллеров: Atmel, STM32, еще x86: MMX/SSE. Еще немного C/C++: нужен boost? нужен QT? ну ладно берем boost, берем QT. Ага… попутно где-то кому-то делаю программу для планшета Android (Java+JNI+cpp) в Eclipse, и еще собираю систему на кристалле в ПЛИС Altera — тут уже язык Verilog, тестбенчи и прочие хардверные радости.

И знаете что? Я уже не уверен, что полностью понимаю как работают все эти системы, программы, алгоритмы. От былой уверенности уже нет и следа. По моим подсчетам в год приходилось и приходится изучать, знакомиться и использовать до 2-3 новых технологий, сред программирования, библиотек и языков. При этом, конечно, нет никакой уверенности, что все знаешь и все понимаешь. На самом деле понимаешь только базовые принципы…

Более того, если когда-то, лет пять назад, я думал, что я знаю, люблю и понимаю c++, то теперь, уже с приходом новых стандартов типа c++11, c++14/17, мне приходится действительно тяжко. Мои коллеги исправляют мой код заменяя мои типизированные переменные на тип auto… И что? Неужели после этого код становится действительно более читаемым? Или вот еще не очень свежая новость: нежелательно использовать наследование, но предпочтительно использовать темплейты. И мне грустно. Я не люблю темплейты.

Я бы вероятно совсем впал в уныние, однако, в частных разговорах с другими программистами за чашечкой коньяка вдруг выясняется, что не только у меня такие проблемы, не один я такой отсталый (хотя не все в этом сразу признаются, зависит от размера чашки). Многие так же, как и я, не до конца понимают чужой код, не до конца понимают как работает вся наша система целиком, не знают, какие сторонние библиотеки мы уже подключили и используем, не понимают откуда в нашей не очень сложной программе 67 потоков и почему она занимает в памяти 350 мегабайт.

Более всего в этой истории с высокоуровневым программированием меня тревожит «эффект магической строки». Этот термин я сам недавно придумал. Речь не о багах, которые конечно были, есть и будут. Речь об одной строке кода, которая принципиально меняет свойства программы.

Например, пишем программу с использованием библиотек QT. У нас есть наш собственный класс, унаследованный от

QGraphicsView. Мы используем наш класс, размещаем на нем QGraphicsScene, добавляем всякие QGraphicsItem и живем долго и счастливо, но медленно. Потом вставляем в код одну магическую строку:

```
QGLWidget* gl = new QGLWidget(); this->setViewport(gl);
```

И вот уже наше приложение зашевелилось и вообще стало довольно шустрым, ведь теперь оно использует для отрисовки графики аппаратное ускорение OpenGL.

> — Эй! Коллега программист! Ты знаешь, что такое OpenGL?
> — Слышал, но никогда не использовал.
> — Я то же. Но теперь мы используем его и это работает! Смотри как хорошо стало!
> — Ура! А ведь всего лишь одна строка кода… ну и две-три дополнительных DLL в инсталляторе.

Проходит неделя или две. Слушайте, чего-то как-то не очень. Какие-то проблемы с медиа плеером и QGraphicsVideoItem… Надо что-то с этим сделать… Еще неделя и появляется еще одна магическая строчка:

```
QCoreApplication::setAttribute(Qt::AA_UseOpenGLES);
```

> — Эй! Коллега программист! Это что еще за хрень?
> — Ну… я не знаю точно, но вроде бы после этого используется режим трансляции запросов OpenGL в DirectX (Angle) и якобы теперь стало гораздо лучше: и CPU не так загружен и видео

> кажется стало без ошибок крутить…
> — ???????????..
> — Ладно, посмотрим, что скажет QA..

Беда N1: программисты до конца не понимают как работает их программа! А есть ли способ понять? Сколько на это потребуется времени? А как же time-to-market? Не наступила ли уже «технологическая сингулярность», когда новые технологии рождаются быстрее, чем мы можем их переварить и освоить?

История третья: Пропасть непонимания

Года три назад посчастливилось нашей компании ввязаться в очень перспективный проект. Мы так думали, что проект очень перспективный. Российская компания по производству оборудования решила снабжать свои агрегаты платой электронного управления. Нам предстояло в сжатые сроки разработать и изготовить плату и плюс программное обеспечение микроконтроллера.

Сказать по правде, я был очень рад этому проекту. Во-первых, компания российская, некий эффект патриотизма. Во-вторых, казалось, что проект весьма долговременный, поскольку ассортимент агрегатов широкий, а заказчик имел намерение идти в ногу со временем и выпускать конкурентное современное оборудование.

Однако были и нюансы:

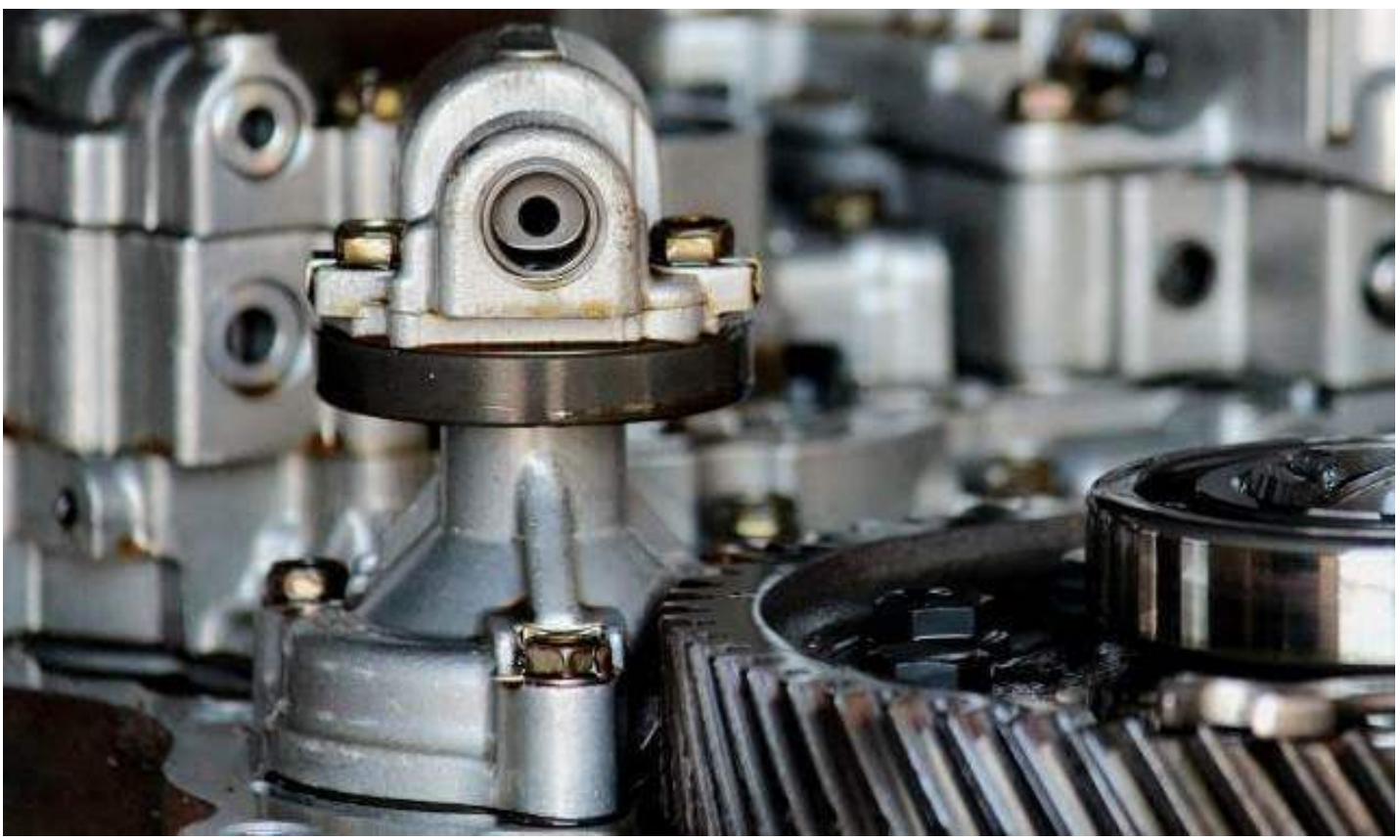
- Заказчик абсолютно не представлял себе, что такое ПО микроконтроллера и как производятся платы. Были даже разговоры, что мол если будет нужно они сами купят линию по монтажу плат и будут сами их комплектовать и производить. Вот это масштаб!

- Мы в свою очередь не очень представляли, как работает их оборудование, ну и вообще тема управления оборудованием для нас была довольно новой. Техническое задание по алгоритму работы платы контроллера предполагалось будет передано нам после подписания договора по началу работ, мы его еще должны согласовать.

Мы почему-то сразу подумали, что возьмем денег по минимуму, все равно по проекту еще долго взаимодействовать и сопровождать его. Заказчик в лице директора с этим устно согласился.

Далее начались некоторые странности. Как оказалось, со стороны заказчика был только один инженер, который реально вникал в

алгоритм, сам его описывал, часто звонил и интересовался ходом работ. При этом, мне постоянно казалось, что мы делаем что-то не то, в наших алгоритмах явно не хватало математики, здесь явно не хватает всех этих ТАУ, ПИД и прочих премудростей систем автоматического управления. Мы пытались развернуть разработку в сторону реального ТАУ, но получили ответ, что получится слишком сложно и им такое не надо. Самое странное, что кроме одного инженера, с которым мы взаимодействовали со стороны заказчика никого более никакие алгоритмы вообще не интересовали. Уже в момент сдачи работ, при демонстрации готовых плат мне наконец-то пришло понимание происходящего. Главный инженер проекта воспринимает плату микроконтроллера, как любую другую деталь агрегата, только что выточенную на токарном станке или согнутую на трубогибе.



То есть: должен быть чертеж с указанием типа материала,

размеры, допуски — все. Есть деталь — устанавливаем ее на посадочное место и весь разговор.

Ну что сказать. Я пытался разговаривать и объяснять, что программное обеспечение — это не деталь, что требуется долгое и тщательное тестирование алгоритмов на реальном оборудовании. Что может оказаться, что реализованные алгоритмы могут потребовать корректировки или изменения постоянных времени, скорости реакции и т.д. Моя фраза, что в программном обеспечении возможны баги, которые нужно отлавливать и устранять наткнулась на стену непонимания. Оказалось, что наши баги — это наши проблемы. ВСЕ баги, все 100%, должны быть устранены до момента сдачи работ. Потом инженер заказчика проводит свое тестирование и приемку работ. И все. Проект сдан, никаких более изменений в ПО не предвидится и изделие после установочной партии передается в серию.

Стоит ли говорить, что сторона заказчика провела испытание плат одним инженером за один (!) день и уже через неделю после подписания акта приемки сдачи он же звонил и просил подправить два временных параметра в алгоритме.

Душа болит и сердце плачет. Мы реально вкладывались в этот проект, жили им, но мы так и не поняли, сделали ли мы всю работу так как надо. Мы так и не увидели своими глазами агрегат в сборе под управлением нашей платы.

Закончилось все так же внезапно, как и началось. По прошествии некоторого времени заказчик рассчитал, что производство

печатных плат в РФ экономически не выгодно. Проект закрыт. Может оно и к лучшему, кто знает…

Отвлекусь от конкретного этого описанного выше случая. Хочу отметить важное: каждый программист в принципе знает, что в его программе есть или возможны ошибки. Да, мы пишем тесты, ведем continuous integration, но у многих из нас в баг трекере висит список незакрытых задач даже по продукту который давно уже выпущен. Иногда программа доходит до этапа «смертельная петля тестирования». Это когда свежие исправления в программе порождают новые, ранее не известные баги. QA возвращает на доработку ПО, разработчики исправляют и отдают в QA, но те, работая не покладая рук возвращают на доработку снова и снова. Если заказчик этого ПО адекватный, то он просто готов смириться с некоторыми ошибками, которые иногда объявляются фичами (feature), с целью снизить напряженность отношений.

С другой стороны, сторонний заказчик, особенно если ранее не сталкивался с процессом разработки ПО, понимает процесс совсем по другому. Для очень многих программа — это то, что можно увидеть и пощупать, то есть как правило — это только часть GUI, то что можно увидеть глазами: кнопочки, формы, диалоги. Если есть плата, то важно, как она выглядит и как ее можно подержать в руках. Что и как за этим скрыто, какая там есть логика и какие алгоритмы — неведомо. Отсюда барьер не понимания.

Беда N2: Заказчик почти никогда не в состоянии проверить качество разработанного программного обеспечения, даже если ему отдать исходные тексты.

Если теперь рассмотреть эти две описанные мною беды как единое целое: программисты не до конца понимают как работают их программы, плюс заказчик не может проверить результат, то что это вообще будет? Какое будущее нас ждет? Сколько недо-продуктов будет нас окружать? Опасно ли входить в высотные здания, проезжать по мостам, рассчитанным в инженерных CAD? Будем ли мы бояться летать на самолетах и ездить в беспилотных автомобилях? Можно ли верить медицинским приборам? Не побоимся ли мы входить в систему онлайн-банка? Программные продукты окружают нас повсюду. Возможно, не будь я программистом, мне не было бы так страшно…

Основные ошибки при открытии небольшого Интернет-магазина 33

Centrobit • 18 июня 2012 в 15:15

Как уменьшить вероятность ошибки на этапе написания кода. Заметка N2 57

Andrey2008 • 29 марта 2011 в 09:58

Intel IPP Samples for Windows — работа над ошибками 15

Andrey2008 • 27 января 2011 в 14:43

Популярное за сутки

Наташа — библиотека для извлечения структурированной информации из текстов на русском языке 14

alexkuku • вчера в 16:12

Unit-тестирование скриншотами: преодолеваем звуковой барьер. Расшифровка доклада 4

lahmatiy • вчера в 13:05

Люди не хотят чего-то действительно нового — они хотят привычное, но сделанное иначе 25

ПЕРЕВОД

Smileek • вчера в 10:32

Руководство по SEO JavaScript-сайтов. Часть 2. Проблемы, эксперименты и рекомендации

2

ПЕРЕВОД

ru_vds • вчера в 12:04

Как адаптировать игру на Unity под iPhone X к апрелю

0

P1CACHU • вчера в 16:13

Лучшее на Geektimes

Стивен Хокинг, автор «Краткой истории времени», умер на 77 году жизни

33

HostingManager • вчера в 13:49

Обзор рынка моноколес 2018

70

lozga • вчера в 06:58

«Битва за Telegram»: 35 пользователей подали в суд на ФСБ

40

alizar • вчера в 15:14

Стивен Хокинг и его работа — что дал ученый человечеству?

8

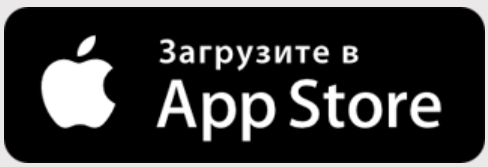
marks • вчера в 14:46

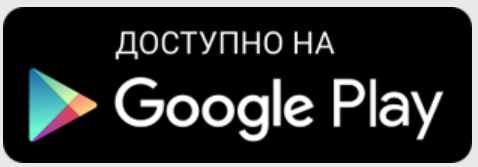
Sunlike — светодиодный свет нового поколения

17

AlexeyNadezhin • вчера в 20:32

Мобильное приложение





Полная версия