Slide 2: О чем эта лекция

Я не собираюсь читать вам лекцию на тему, что надо проверять возвращаемые из malloc значения, проверять форматные строки в printf и т.д. и т.п. Вы и так уже это все слышали миллион раз, и повторение в миллион первый вряд ли что изменит. Вместо этого я немного расскажу к чему это все приводит, а также поделюсь некоторыми не столь известными сведениями о том, какие бывают ошибки, что с ними делать, чего иногда лучше не делать. Ну и еще немного мы коснемся смежных областей - там тоже бывают ошибки.

А теперь немного истории. В английском языке для термина 'ошибка' есть 2 разных слова - это слово error (оно обозначает ошибку в самом общем смысле), а есть слово 'bug' - это вполне конкретная разновидность программных ошибок.
Откуда же пошел этот термин?

Slide 3&4: Что такое ‘bug’?

9 Сентября 1947г. Гарвардский университет. Журнал обслуживания компьютера Марк II. Уже ставшая классикой строка с прилепленной скотчем молью, отметкой времени 15:45 и записью - "Реле №70, Панель F (моль) в реле" и гордая пометка внизу - "Первый случай реального бага"

Очевидно, что термин 'bug' зародился не тут - его бы не использовали в сочетании 'реальный баг'. Этот термин появился более чем за 100 лет до истории о моли-в-реле.

<Enter>

Даже Томас Эдисон использовал этот термин в письме Теодору Пускасу (цитата из 'Ельского сборника цитат' 2006г):

'Bugs' -- так называют маленький сбой или затруднение -- покажет себя, и могут потребоваться месяцы интенсивных наблюдений, изучений и упорной работы, перед тем как достигнуть безусловного успеха или неудачи.

Это слово давным-давно использовалось для обозначения Монстров, и те, кто проводил недели и месяцы за отладкой своего (а тем паче чужого) кода, должны это очень хорошо понимать.

Со времен Марка II bug'и несколько подросли, и теперь это уже не моль, а нечто такое - <Enter>

Итак, давайте познакомимся с классикой bug'ов. Нечто по слухам, хорошо известное, но почти никто не знает реальной истории.

Slide 5: Mariner 1

22 Июля 1962г, первый космический корабль NASA Mariner отправился с миссией на Венеру. Он счастливо стартовал с Мыса Канаверал, но буквально через несколько минут начал уходить с курса. Системе управления не удалось исправить траекторию, и наземная команда так же не смогла исправить ее вручную.

Так как ракета свернул в сторону Североатлантической океанской трассы, офицер, отвечающий за безопасность, сделал единственное, что он мог сделать: взорвать аппарат. Через четыре минуты и 55 секунд после начала миссии Mariner 1 был взорван.

NASA уже и так по черному завидовала СССР с его Спутником, и инцидент с Mariner стал еще одним международным позором для агентства. Расследование этой катастрофы показало то, что НАСА назвала "неправильной эксплуатации маяков оборудования Atlas» - хотя позже выяснилось, что реальной причиной была ошибка в одном знаке пунктуации, что вызвало фатальную ошибку в программном обеспечении.

По широко распространенному мнению это была замена запятой на точку в операторе DO FORTRAN'a, но на самом деле это была черта над знаком радиуса (точнее производной от радиуса, т.е. скорости). Черта эта обозначала оператор усреднения. Программист, переносивший уравнения из тетради, где они были написаны, либо не обратил внимания на этот знак, либо вообще не знал, что это такое.

В результате система управления получилась крайне не устойчивая, и малейшие колебания скорости (так как усреднение отсутствовало) вызывали немедленную мощную реакцию со стороны корректирующих двигателей.

Результат - $18.2M отправились на ветер.

Это была далеко не последняя авария в космосе, связанная с софтом. Иногда причины аварий были просто анекдотичными.

Slide 6: *Mars Climate Orbiter*

Аппарат для исследования климата Марса (Mars Climate Orbiter) является частью программы Mars Surveyor по изучению и картографированию Марса. Ожидалось, что программа Mars Surveyor продлится в течение 10 лет и в рамках программы будет запускаться одна экспедиция в год. Первыми двумя экспедициями были миссии аппаратов *Mars Pathfinder* и *Mars Global Surveyor* в 1996 году. Аппарат *Mars Climate Orbiter* был запущен 11 декабря 1998 года. Следом за ним был также запущен *Mars Polar Lander* - 3 января 1999. Оба аппарата были потеряны вскоре после того, как они достигли красной планеты. Эти два космических корабля стоили NASA около 327,6 миллиона долларов, потраченных на их создание и функционирование. Эти аварии заставили NASA пересмотреть свои цели и методы в Марсианской программе, чтобы быть уверенными в успехе будущих миссий. Причину аварии *Mars Polar Lander* определить все еще нельзя. Однако причина потери *Mars Climate Orbiter* выяснена, поэтому мы сконцентрируем наше исследование на этом аппарате.

Mars Climate Orbiter был запущен 11 декабря 1998 года с помощью ракеты-носителя *Delta 11* с космодрома на мысе Канаверал (Canaveral) во Флориде. После девяти с половиной месяцев космического полета 23 сентября 1999 года, его планировалось вывести на орбиту вокруг Марса. Однако, когда пришло назначенное время, что-то произошло. Около 2:00 утра по летнему тихоокеанскому времени, аппарат включил главный двигатель для выхода на орбиту вокруг планеты. Запуск двигателя начался как планировалось, за пять минут до того, как аппарат оказался за планетой (если смотреть с Земли). Управление полетом не зафиксировало сигнала, когда ожидалось, что аппарат должен был выйти из-за планеты[35].

Mars Climate Orbiter разрабатывался объединенной командой инженеров и ученых в двух местах - Лаборатории реактивных двигателей (Jet Propulsion Laboratory, JPL), расположенной в Пасадене (Pasadena), штат Калифорния, и на заводе Lockheed Martin Astronautics, LMA в Денвере, штат Колорадо. Завод LMA был ответственен за планирование и разработку *Mars Climate Orbiter* с точки зрения интеграции и тестирования полетных систем, а также за операцию запуска. Лаборатория JPL несла ответственность за управление проектом, за управление разработкой космического аппарата и приборов, за системотехнику, за планирование миссии, навигацию, разработку операционной системы миссии, разработку наземной системы сбора данных и гарантии безопасности[36].

В ходе девяти с половиной месяцев полета наземные службы отслеживали и сравнивали наблюдаемую траекторию аппарата с расчетной. Также наземные службы проверяли все события, происходящие на борту *Mars Climate Orbiter*. Одним из таких событий было уменьшение углового момента (Angular Momentum Desaturation, AMD). Событие AMD возникало, когда аппарат запускал двигатели малой тяги для устранения углового момента, накопившегося в его маховиках. В основе своей - это калибровочный маневр для поддержания функционирования системных компонентов в заданном диапазоне. Когда возникало событие AMD, происходила следующая последовательность событий:

1. Аппарат посылал значимые данные на наземную станцию.
2. Данные обрабатывались программным модулем, называвшимся SM-FORCE.
3. Результаты работы модуля SM-FORCE помещались в файл, называемый файл AMD.
4. Данные файла AMD использовались для вычисления изменения скорости аппарата.
5. Вычисленное значение изменения скорости использовалось для моделирования траектории корабля.

Согласно спецификации модуль SM-FORCE должен формировать данные, помещаемые в файл AMD, используя метрические единицы, то есть ньютон-секунды. Однако по тем или иным причинам модуль SM-FORCE на наземной станции выводил данные, используя английские единицы (фунт-секунды) (в официальном отчете причина не указывалась). Программный модуль, который рассчитывал изменение скорости с использованием данных из файла AMD, ожидал, что они будут в метрических единицах, согласно спецификации. На борту аппарата модуль, который создавал файл AMD, использовал метрические единицы. Это привело к различию между траекториями, вычисленными космическим аппаратом и наземной станцией, а именно параметры траектории, вычисленные наземной станцией, были в 4,45 раза меньше, поскольку 1 фунт-секунда равен 4,45 ньютон-секундам.

Корабль периодически передавал вычисленную модель траектории на наземную станцию для сравнения. Теоретически быстрое сопоставление моделей, полученных кораблем и наземной станцией, должно было поднять тревогу. Однако несколько осложняющих факторов помешали наземному персоналу осознать ошибку.

- В программном обеспечении наземной станции было несколько ошибок, и персонал не мог использовать модуль SM-FORCE для расчета траектории корабля. Эти ошибки были исправлены только к четвертому месяцу полета, в районе апреля 1999 года.
- Персонал, ответственный за навигацию, не знал о том, что данные об изменении скорости с борта корабля были доступны для сравнения в течение долгого времени после запуска.
- Линия обзора между аппаратом и Землей не давала персоналу точно моделировать траекторию корабля, используя наблюдения.

Если бы событие AMD возникало нечасто, коэффициент 4,45 мог и не иметь таких серьезных последствий. Однако из-за формы корабля это событие возникало в 10-14 раз чаще, чем ожидалось. Более того, когда были обнаружены различия в моделях наземной станции, космического корабля и данных наблюдений, неофициальный отчет об этих различиях был отправлен по электронной почте, без использования стандартной процедуры для таких случаев. В конечном итоге эти различия не были устранены до потери космического аппарата.

8 сентября 1999 года персонал наземной станции рассчитал маневр для вывода корабля на орбиту Марса. Это вычисление было сделано с использованием неверной модели. Целью этого маневра была коррекция траектории корабля таким образом, чтобы точка наибольшего приближения к Марсу составляла 226 километров. 23 сентября маневр был выполнен. Запуск двигателя произошел в 09:00:46 по Всемирному координированному времени (UTC). Спустя четыре минуты и шесть секунд наземная станция потеряла сигнал с аппарата. Потеря сигнала была вызвана тем, что Orbiter находился за планетой. Но это событие произошло на 49 секунд раньше, чем было предсказано моделью траектории. Поскольку была использована ошибочная модель траектории, корабль оказался в действительности ближе к поверхности планеты, чем ожидалось. Действительная точка максимального сближения оказалась приблизительно 57 километров. По оценке, такая высота оказалась слишком мала для аппарата.

Стоимость этой ошибки составила более $125M

Иногда ошибки возникают там, где их никто не ждал. Иногда разные программные модули системы начинают взаимодействовать неожиданным образом.

Slide 7: Конвертоплан Osprey

За пару недель до рождества 2000г, на Конвертоплане Osprey - гибрид самолета и вертолета, произошла авария гидравлической системы, которая, по замыслу конструкторов, должна была закончится без последствий. В случае аварии гидравлики на одном из двигателей, Osprey должен был перейти в режим вертолета и посадить машину.

Однако, согласно отчету по катастрофе, была проблема в "аномалии программного обеспечения". Когда компьютер обнаружил аварию в гидравлической системе, он остановил вращение винтов.

Пилот стал проводить обычную процедуру и нажал кнопку сброса, что бы включить двигатели. В этот момент обе опоры винтов испытали "существенные изменения тангажа и тяги" и в конце концов застряли. Самолет упал в болото, все 4 морских пехотинца на борту погибли.

Что же произошло на борту до сих пор неизвестно: производители Osprey Boeing и Bell Helicopter заявили, что единственное, что было изменено - это программное обеспечение. Правительство затребовало подробностей, но они так и не последовали.

Иногда ошибки бывают не столь значительными по последствиям, но гораздо более неожиданными и загадочными. Причем чисто аппаратные ошибки тоже бывают весьма интригующими. Поиск таких ошибок вполне можно рассматривать как работу для детектива …

Slide 8: Неожиданные ошибки

Not enough memory в программе не использующей память
Было это в конце 80х, когда самой крутой PC была '386я с сопроцессором'. И была написана маленькая программа, которая что то считала и выводила несложную графику (все в DOS). У меня все работало замечательно, а вот у заказчика оно свалилось с сообщением 'Not enough memory' (что было очень странно, т.к. этой самой memory ей было нужно совсем не много). Попытка взять машину с памятью побольше успехом не увенчалась. Раскопки с TurboDebugger'ом показали удивительную вещь:
Программа использовала вычисления с плавающей точкой для расчета размеров (и площади) фигур, которые она пыталась нарисовать. По подсчитанной площади делался malloc - для выделения памяти под растр. В машине отсутствовал сопроцессор, а в программе отсутствовала проверка на его наличие. В результате все вычисления с плавающей точкой были проигнорированы, и в malloc свалился 0, что и вызвало выдачу NULL и как следствие ругань на нехватку памяти.

Magic-More Magic выключатель на PDP 10

Когда-то, в одной из лабораторий MIT'а стояла машина PDP-10. На ней обнаружился кустарно приделанный выключатель, кто его приделал - никто не знал.

Как вы понимаете неизвестные штуки на компьютере лучше не трогать, вдруг компьютер зависнет. Выключатель был помечен весьма необычным образом. Он имел 2 позиции, и на нем самом карандашом было нацарапано - Magic и More Magic. Выключатель стоял в позиции More Magic.

Позвали еще одного хакера, что бы он посмотрел на это. Он сказал, что никогда не видел выключателя раньше. Более подробное исследование выключателя показало, что от него отходил только 1 провод. Другой конец этого провода исчезал где то в лабиринте проводов внутри компьютера, но исходя из самых базовых принципов электричества было абсолютно понятно, что выключатель не может делать ничего, пока к нему не подсоединят как минимум 2 провода. У этого выключателя провод был только с одной стороны, с другой стороны ничего не было.

Было ясно, что этот выключатель - чья то глупая шутка. Придя к выводу, что выключатель явно не рабочий, его переключили. Компьютер немедленно повис.

Придя в себя, экспериментаторы заключили, что это было совпадение, но тем не менее вернули переключатель в More Magic позицию перед перезагрузкой компьютера.

Год спустя эту историю рассказали еще одному хакеру, вроде это был David Moon. Он явно усомнился в здравом уме рассказчика, или заподозрил его в вере в сверхъестественную силу выключателя, или может быть подумал, что его дурят, рассказывая нелепые истории. Что бы доказать ему, пришлось показать тот самый выключатель, который все еще был прилеплен к стойке машины с тем же самым одним проводом, и в той же самой позиции More magic. Они тщательно исследовали выключатель и его одинокий провод, и обнаружили, что его второй конец, хотя и был подсоединен к проводке компьютера, был подсоединен к выводу заземления. Это явно и недвусмысленно делало выключатель дважды бесполезным - он не только электрически не мог работать на одном проводе, этот провод был подключен туда, где он ни на что не мог повлиять. Так что они переключили выключатель. … Компьютер немедленно повис.

На этот раз они пошли к Richard Greenblatt, ветерану MIT, он был неподалеку. Он тоже никогда не замечал этого выключателя раньше. Он его исследовал, сделал вывод, что выключатель бесполезен, извлек кусачки и выдрал этот выключатель вместе с проводом. Затем перезагрузил компьютер. Компьютер нормально заработал, и продолжает работать до сих пор.

До сих пор неизвестно, как именно этот выключатель подвешивал машину. Возможно, какая то цепь около точки подключения была очень чувствительна, и переключения было достаточно для изменения емкости, что бы какая то много мегагерцовая цепь перестала работать. Но точно этого уже не узнать, так что можно просто сказать, что выключатель просто был магическим.

В 1994г было предложено еще одно объяснение этой истории - корпус выключателя был металлическим, и вполне возможно, что один из контактов выключателя (тот самый, неподсоединенный) был соединен с корпусом. В таком случае выключатель просто соединял корпус машины с определенной точкой в земляной цепи. Так как размеры корпуса, да и самой земляной цепи были весьма большими, между ними вполне могла образоваться разность потенциалов. В таком случае замыкание выключателя могло вызвать помехи в цепи земли, которые вполне могли бы повесить машину.

ЭВМ с ‘фотоэффектом’

Это было где-то в 86-87 годах в МФТИ. Один студент собрал себе компьютер (Радио 86РК). Это был небольшой компьютер, собранный вокруг советского аналога процессора i8080 и некоторых микросхем из его периферийного набора. В качестве дисплея использовался телевизор, изображение на котором формировалось контролером CRT (i8275). Так же в компьютере был контролер DMA (i8257), который обеспечивал доставку данных из памяти для показа на экране.

Компьютер был полностью собран, отлажен, и все работало, как часы. До тех пор, пока не попробовали закрыть крышку корпуса. Изображение с экрана телевизора немедленно пропало, а компьютер завис.

Сначала грешили на то, что крышка где то что то придавила, но ничего такого обнаружено не было. Тогда вооружились осциллографом, и стали скрупулезно исследовать сигналы на выводах микросхем. Обнаружили интересную аномалию - если один из проводов исследовали осциллографом с одного конца, там были правильные красивые сигналы, если же осциллографом касались ножки микросхемы с другого конца провода, то сигнал немедленно пропадал (вместе с изображением на экране). Провод был абсолютно целый - его сначала прозвонили тестером, а потом пытались испытать на разрыв.

Дальнейшие исследования показали, что этот провод не один такой - практически любой провод в районе этой микросхемы (а был это тот самый контролер DMA) вел себя именно так. Это было весьма неожиданное поведение.

Далее было еще неожиданнее - оказалось, что даже касаться щупом осциллографа ножки микросхемы не обязательно - просто поднести щуп в район этой микросхемы (не касаясь ничего) уже достаточно.

Дальнейшие исследования показали, что даже щуп осциллографа не нужен - достаточно поднести туда руку. Закрались подозрения о наводках от руки, но они не оправдались - наводок не было.

Наконец причина была найдена - на столе стояла осветительная лампа и светила на место наших экспериментов, в том числе доставалось и микросхеме DMA. Причина была весьма и весьма неожиданна, поэтому был поставлен контрольный эксперимент. Деревянной линейкой закрывали свет от лампы, и когда тень касалась того места, где у микросхемы был расположен кристалл, машина висла.

Дальнейшие исследования показали, что на плате был обрыв в проводе, по которому в микросхему регистров защелкивалась старшая половина адреса от DMA контролера. И регистр срабатывал исключительно из-за наводок в цепях питания и земли. Ближайшая микросхема была как раз микросхемой DMA, и именно от ее наводок адрес и защелкивался. Когда переставали светить на кристалл контролера DMA, уровень наводок снижался, и регистр переставал на них реагировать.

Сброс ПО над Мертвым морем

Фирма Motorola испытывала новый процессор для автопилота на истребителе в Израиле. Все было отлажено. Пилоты на испытаниях отправились «огибать рельеф» с севера до юга Израиля. Истребитель прекрасно пролетел на автопилоте над равнинной частью, над горной частью, над долиной реки Иордан, и приближался к Мёртвому морю. Не долетев до него, неожиданно происходит общий сброс процессора, автопилот выключается на полном ходу, пилоты переходят на ручное управление, и сажают истребитель.

Процессор отправили на доработку и тестирование. Все тесты прошли снова без сбоев. Снова начали реальную проверку. Истребитель пролетел над всеми территориями, но при подлете к Мёртвому морю: общий сброс, выключение автопилота, ручная посадка.

Длительные тесты не могли выявить никаких изъянов. После продолжительных попыток было найдено, что программы автопилота при вычислении параметров управления по глубоко научным секретным формулам производили деление на значение текущей высоты истребителя над уровнем океана. При подлете к Мёртвому морю, высота становилась нулевой, и процессор при делении на ноль давал общий сброс. До этого случая никому не приходило в голову, что самолеты могут летать ниже уровня океана...

F22 и «лини перемены дат»
При попытке перегнать истребители F22 «своим ходом» с Гавайских островов на базу ВВС Kadena на японском острове Окинава программный сбой «в навигационном обеспечении» вынудил пилотов развернуться и возвратиться туда, откуда вылетели. Теперь стала известна истинная природа «навигационной аномалии».

Как сообщает DefenseTech со ссылкой на Associated Press, истребители не сумели преодолеть так называемую «линию перемены дат» - условную линию, по разные стороны которой местное время одно и то же (с точностью до часового пояса), но календарные даты различаются на одни сутки. Линия перемены дат проходит по меридиану 180 градусов с отдельными отклонениями.

Перемена дат осуществляется (и вообще имеет смысл) лишь при использовании местного времени. При пересечении линии перемены дат необходимо либо прибавлять, либо вычитать одни сутки – в зависимости от того, в каком направлении осуществляется движение. По всей видимости, этот парадокс Земного шара, осознанный еще участниками экспедиции Магеллана, был позабыт разработчиками F-22 Raptor.

Последствия такой забывчивости оказались весьма ощутимыми. У истребителей в полете, отмечает Defense Tech, вышли из строя топливная и навигационная системы, а также частично связь.

Лишь одному пилоту удалось связаться с экспертами разработчика (компании Lockheed Martin). Несколько пилотов попытались перезагрузить ПО истребителя в полете.

Победить ошибку не удалось, однако сами истребители и их пилоты уцелели, что в подобной ситуации следует считать несомненной удачей. Возвращение на Гавайские острова потребовало дополнительной дозаправки в воздухе.

Впоследствии «навигационную аномалию» удалось исправить, и F-22 все-таки прибыли на авиабазу назначения.

F16 и экватор

Испытания американского истребителя F-16 проводились, понятное дело, в северном полушарии. На заключительном этапе самолет решили проверить где-то в Латинской Америке, но уже с другой стороны экватора. При переводе самолета в режим автопилота он автоматически развернулся "вверх ногами".

Slide 9: Неожиданные ошибки (Необычный источник настройки)

В одной в.ч. была аппаратура, которая принимала от другой связной аппаратуры команды (а именно частоту настройки приёмника), и настраивала этот самый приёмник. Мы же занимались тем, что заменяли первую аппаратуру на свою, сделанную уже на компьютерах. Проблема же заключалась в том, что при смене частоты, я должен был сбросить предыдущую, установить новую, записать новую в БД, пискнуть PC-speaker'ом и много чего еще... а частота сменялась след. образом

мало того, иногда один из разрядов "падал" в 0 на некоторое время, а потом "восстанавливался"

Slide 10: Ariane 5

4 июня 1996 года был произведен первый запуск ракеты-носителя Ariane 5 - детища и гордости Европейского Сообщества. Уже через неполные 40 сек. все закончилось взрывом. Автоподрыв 50-метровой ракеты произошел в районе ее запуска с космодрома во Французской Гвиане. За предшествующие годы ракеты серии Ariane семь раз терпели аварии, но эта побила все рекорды по вызванным ею убыткам. Только находившееся на борту научное оборудование потянуло на пол-миллиарда долларов, не говоря о прочих разнообразных издержках; а астрономические цифры "упущенной выгоды" от несостоявшихся коммерческих запусков и потеря репутации надежного перевозчика в очень конкурентном секторе мировой экономики ("стоимость рынка" к 2000 г. должна превысить 60 млрд. долл.) с трудом поддаются оценке. Небезынтересно отметить, что предыдущая модель - ракета Ariane 4 - успешно запускалась более 100 раз.

Технические подробности аварии

Положение и ориентация ракеты-носителя в пространстве измеряются Навигационной Системой (Inertial Reference Systems - IRS), составной частью которой является встроенный компьютер, вычисляющий углы и скорости на основе информации от бортовой Инерциальной Платформы, оборудованной лазерными гироскопами и акселерометрами. Данные от IRS передаются по специальной шине на Бортовой Компьютер (On-Board Computer - OBC), который обеспечивает необходимую для реализации программы полета информацию и непосредственно - через гидравлические и сервоприводы - управляет твердотопливными ускорителями и криогенным двигателем типа Вулкан (Vulkain).

Как обычно, для обеспечения надежности Системы Управления Полетом используется дублирование оборудования. Поэтому две системы IRS (одна - активная, другая - ее горячий резерв) с идентичным аппаратным и программным обеспечением функционируют параллельно. Как только бортовой компьютер OBC обнаруживает, что "активная" IRS вышла из штатного режима, он сразу же переключается на другую. Впрочем, и бортовых компьютеров тоже два.

Что произошло:

- в момент H0+39 сек. из-за высокой аэродинамической нагрузки вследствие превышения "углом атаки" критической величины на 20 градусов произошло отделение стартовых ускорителей ракеты от основной ее ступени, что и послужило основанием для включения Системы Автоподрыва ракеты;
- изменение угла атаки произошло по причине нештатного вращения сопел твердотопливных ускорителей;
- такое отклонение сопел ускорителей от правильной ориентации вызвала в момент H0 + 37 сек. команда, выданная Бортовым Компьютером на основе информации от активной Навигационной Системы (IRS 2). Часть этой информации была в принципе некорректной: то, что интерпретировалось как полетные данные, на самом деле являлось диагностической информацией встроенного компьютера системы IRS 2;
- встроенный компьютер IRS 2 передал некорректные данные, потому что диагностировал нештатную ситуацию, "поймав" исключение (exception), выброшенное одним из модулей программного обеспечения;

- при этом Бортовой Компьютер не мог переключиться на резервную систему IRS 1, так как она уже прекратила функционировать в течение предшествующего цикла (занявшего 72 мсек.) - по той же причине, что и IRS 2;
- исключение, "выброшенное" одной из программ IRS, явилось следствием выполнения преобразования данных из 64-разрядного формата с плавающей точкой в 16-разрядное целое со знаком, что привело к "Operand Error";
- ошибка произошла в компоненте ПО, предназначенном исключительно для выполнения "регулировки" Инерциальной Платформы. Причем - что звучит парадоксально, если не абсурдно - этот программный модуль выдает значимые результаты только до момента H0 + 7 сек. отрыва ракеты со стартовой площадки. После того, как ракета взлетела, никакого влияния на полет функционирование данного модуля оказать не могло;
- однако, "функция регулировки" действительно должна была (в соответствии с установленными для нее требованиями) действовать еще 50 сек. после инициации "полетного режима" на шине Навигационной Системы (момент H0-3 сек.), что она с усердием дурака, которого заставили богу молиться, и делала;
- ошибка "Operand Error" произошла из-за неожиданно большой величины BH (Horizontal Bias - горизонтальный наклон), посчитанной внутренней функцией на основании величины "горизонтальной скорости", измеренной находящимися на Платформе датчиками. Величина BH служила индикатором точности позиционирования Платформы;
- величина BH оказалась много больше, чем ожидалось потому, что траектория полета Ariane 5 на ранней стадии существенно отличалась от траектории полета Ariane 4 (где этот программный модуль использовался ранее), что и привело к значительно более высокой "горизонтальной скорости".

Финальным же действием, имевшим фатальные последствия, стало прекращение работы процессора; соответственно, вся Навигационная Система перестала функционировать. Возобновить же ее действия оказалось технически невозможно.

- **Slide 11: Что надо делать не всегда**

Переиспользование программных модулей
Модуль калибровки "Инерциальной платформы" от Ariane 4 стал причиной краха Ariane 5
И это не единичный пример, дальше мы увидим еще несколько примеров такого же фатального 'переиспользования' ПО

Прежде чем что то переиспользовать, внимательно изучите спецификации и убедитесь, что это возможно. Если спецификации на модуль не полны, или их вообще нет - то лучше такой модуль выкинуть сразу.

Использование ‘надежных’ языков
В Ariane 5 использовался язык Ada, являющийся одним из самых строго типизированных и защищенных языков. Как видим, даже использование таких языков не дает гарантии отсутствия ошибок, а иногда даже наоборот.

Использование exception’ов для сигнализации об ошибках
Как сигнализировать об ошибках? И как их обрабатывать? Может показаться, что механизм исключений (exception'ов) является панацеей, но это далеко не так.
Что бы exception'ы успешно работали, программа должна с самого начала писаться с учетом того, что в любом ее месте могут возникнуть исключения.
Языки сами по себе не обеспечивают безопасную обработку этих самых исключений.

Во первых - если вы не перехватите исключение вообще, то реакция будет мало отличаться от обычного падения программы. Просто вместо сообщения 'segmentation falult' вы получите нечто вроде 'программа потребовала исполняющую систему завершить ее необычным образом'. Лично у меня это сообщение почему то ассоциируется с Камасутрой. А если у пользователя при этом пропадет документ, который он в поте лица редактировал последние 10 часов, то с Камасутрой придется ассоциировать разработчика.

Тут возникает побочный вопрос - где именно перехватывать исключение? Если слишком близко к месту его возникновения, то этих перехватчиков будет добрых пол программы, если поближе к main'у, то у обработчика исключения не будет возможности сделать что-либо осмысленное, кроме как завершить программу с извинениями.

Во вторых - в языках типа C++ пролетающее исключение будет оставлять за собой хвост из потерянных ресурсов, т.к. никто не будет автоматически делать delete к уже сделанным new. Применение различных smart pointer'ов вместо обычных эту проблему решит (отчасти), но это уже и есть то самое 'с самого начала писаться с учетом '. И smart pointer'ы не заменят полноценный сборщик мусора, т.к. они отвратительно обращаются с циклическими структурами. В языках со встроенным сборщиком мусора будет проблема с финализацией объектов при распространении исключения. Деструкторы этих объектов (если они вообще будут поддерживаться языком), будут вызваны далеко не всегда именно в момент распространения исключения.

Ну и в третьих - в интерактивных приложениях выброшенное из конструктора исключение может оставить GUI в полу-созданом состоянии. Оно будет видно на экране, но половина элементов управления не будет работать, а другая половина останется пустой.

Протоколирование ошибок

Убедитесь, что сам процесс протоколирования не нарушает работу других модулей.
В Ariane 5 механизм обработки ошибок предусматривал следующие три основных действия. Прежде всего, информация о возникновении нештатной ситуации должна быть передана по шине на бортовой компьютер OBC (который правда воспринял это сообщение как часть управляющей команды); параллельно она - вместе со всем контекстом - записывалась в перепрограммируемую память EEPROM (которую во время расследования удалось восстановить и прочесть ее содержимое), и наконец, работа процессора IRS должна была аварийно завершиться. Последнее действие и оказалось фатальным - именно оно, случившееся в ситуации, которая на самом деле была нормальной (несмотря на сгенерированное из-за незащищенного переполнения программное исключение), и привело к катастрофе.

Резервное функционирование модулей
Иногда разработчик не отключает во время уже не нужные программные модули. Иногда это действительно имеет смысл, но чаще всего это ведет к ошибкам.

И так - снова Ariane 5

Прежде всего проследим, откуда взялось первоначальное требование на продолжение выполнения операции регулировки после взлета ракеты. Оказывается, оно было заложено более чем за 10 лет до рокового события, когда проектировались еще ранние модели серии Ariane. При некотором (весьма маловероятном!) развитии событий взлет мог быть отменен буквально за несколько секунд до старта, например в промежутке H0-9 сек., когда на IRS запускался "полетный режим", и H0-5 сек., когда выдавалась команда на выполнение некоторых операций с ракетным оборудованием. В случае неожиданной отмены взлета необходимо было быстро вернуться в режим "обратного отсчета" (countdown) - и при этом не повторять сначала все установочные операции, в том числе приведение к исходному положения Инерциальной Платформы (операция, требующая 45 мин. - время, за которое можно потерять "окно запуска").

Было обосновано, что в случае события отмены старта период в 50 сек. после H0-9 будет достаточным для того, чтобы наземное оборудование смогло восстановить полный контроль за Инерциальной Платформой без потери информации - за это время Платформа прекратит начавшееся было перемещение, а соответствующий программный модуль всю информацию о ее состоянии зафиксирует, что поможет оперативно возвратить ее в исходное положение (напомним, что все это в случае, когда ракета продолжает находиться на месте старта). И действительно, однажды, в 1989 году, при старте под номером 33 ракеты Ariane 4, эта особенность была с успехом задействована.

Однако, Ariane 5, в отличие от предыдущей модели, имел уже принципиально другую дисциплину выполнения предполетных действий - настолько другую, что работа рокового программного модуля после времени старта вообще не имела смысла. Однако, модуль повторно использовался без каких-либо модификаций - видимо из-за нежелания изменять программный код, который успешно работает.

Использовать ограниченность входных данных
Иногда разработчики не проверяют входные данные на основании того, что 'они физически не могут быть не такими, как надо'. Такой подход чреват весьма неожиданными ошибками в не менее неожиданными последствиями.
Снова Ariane - почему же произошло переполнение в приведении типа:

Расследование показало, что в данном программном модуле присутствовало целых семь переменных, вовлеченных в операции преобразования типов. Оказалось, что разработчики проводили анализ всех операций, способных потенциально генерировать исключение, на уязвимость. И это было их вполне сознательным решением добавить надлежащую защиту к четырем переменным, а три - включая BH - оставить незащищенными. Основанием для такого решения была уверенность в том, что для этих трех переменных возникновение ситуации переполнения невозможно в принципе. Уверенность эта была подкреплена расчетами, показывающими, что ожидаемый диапазон физических полетных параметров, на основании которых определяются величины упомянутых переменных, таков, что к нежелательной ситуации привести не может. И это было верно - но для траектории, рассчитанной для модели Ariane 4. А ракета нового поколения Ariane 5 стартовала по совсем другой траектории, для которой никаких оценок не выполнялось. Между тем она (вкупе с высоким начальным ускорением) была такова, что "горизонтальная скорость" превзошла расчетную (для Ariane 4) более чем в пять раз.

Но почему же не была (пусть в порядке перестраховки) обеспечена защита для всех семи, включая BH, переменных? Оказывается, для компьютера IRS была продекларирована максимальная величина рабочей нагрузки в 80%, и поэтому разработчики должны были искать пути снижения излишних вычислительных издержек. Вот они и ослабили защиту там, где теоретически нежелательной ситуации возникнуть не могло. Когда же она возникла, то вступил в действие такой механизм обработки исключительной ситуации, который оказался совершенно неадекватным.

Slide 12: Еще несколько примеров

Спецификация на ПО тоже может содержать ошибки (Ariane 5)
Считается, что надежность ПО традиционно определяется степенью его соответствия зафиксированным в спецификациях требованиям; однако, часто бывает так, что ПО делает именно то, что ему и было предписано, и авария Ariane 5 - классический тому пример: и злополучное вычисление посторонней для полета величины горизонтального отклонения Инерциальной Платформы, и реакция на него вплоть до выведения из строя всех навигационных систем и бортовых компьютеров - все это случилось в полном соответствии с Требованиями, которые были частично унаследованы от Ariane 4 и не отражали новых реальностей. Более того, по сравнению с ошибками в коде именно спецификационные ошибки обычно ведут к более тяжелым последствиям - компетенции разработчиков ПО недостаточно для обнаружения таких ошибок. Программный комплекс - сложная система, однако реальный мир, отражаемый в спецификационных требованиях - еще более сложен и требует специальных экспертных знаний.

Запуск печи бойлера с помощью паяльной лампы
Система управления бойлером была выполнена на 8ми битном микроконтроллере, и успешно проработала несколько лет. И вот в один прекрасный зимний вечер в печи бойлера вышла из строя термопара, которая замеряла температуру в камере сгорания. Под рукой новой термопары не оказалось, и за ней отправились в соседний город. Через несколько часов термопару привезли, заменили и попытались запустить бойлер. Система управления наотрез отказалась подавать горючее в топку и вообще как либо работать. Внимательное изучение показало, что температура в печи измеряемая термопарой выдавалась в градусах Цельсия, и сохранялась с целой беззнаковой переменной, которая на этом микроконтроллере была 16ти битная. Так как после остановки бойлера прошло довольно много времени, а на улице было -20 градусов, то термопара именно эту температуру и выдала. После преобразования в беззнаковый вид, там получилось 65тысяч с гаком градусов. Система управления справедливо рассудила, что зачем вам печка, если у вас там и так небольшое солнце.
Разработчик не предполагал, что температура в камере сгорания может стать отрицательной.
Это кстати был единственный случай, когда бойлер пришлось заводить с помощью паяльной лампы (греть камеру сгорания и термопару)

Попытка внедрить в Англии систему управления воздушным движением
Попытка внедрить в Англии программную систему управления воздушным движением, которая до того несколько лет успешно эксплуатировалась в США, оказалась сопряжена с большими трудностями - ряд модулей весьма оригинальным образом обращались с информацией о географической долготе: карта Англия уподоблялась листу бумаги, согнутому и сложенному вдоль Гринвичского меридиана, и получалось, что симметрично расположенные относительно этого нулевого меридиана населенные пункты накладывались друг на друга. В Америке, через которую нулевой меридиан не проходит, эти проблемы никак не проявлялись.

Slide 13 & 14: Therac 25

В 1985-87 гг. 6 человек получили смертельную дозу облучения во время сеансов радиационной терапии с применением медицинского ускорителя Therac-25 (количество пациентов, также подвергшихся переоблучению, но выживших, точно не известно). Производителем установки являлось одно из подразделений Канадского Агентства Атомной Энергии (Atomic Energy of Canada Limited - AECL). Модель Therac-25, законченная в виде прототипа в 1976 г. и поступившая в промышленную эксплуатацию в 1982 г. (пять установок в США и шесть - в Канаде) была развитием ранних моделей Therac-6 и Therac-20. При этом некоторые программные модули, разработанные для ранних моделей, использовались повторно (в том числе - поставленные партнером, французской фирмой CGR, сотрудничество AECL с которой прекратилось к моменту начала работ над Therac-25).

Kennestone Regional Oncologyl Center, г. Кенстоун, округ Мариетта, штат Джорджия (Kennestone, Marietta, Georgia). Июнь 1985

Женщина в возрасте 61 года была направлена в онкологический центр для дополнительного лечения аппаратом Therac-25 после хирургического удаления молочной железы. Как считается, пациентка получила одну или две дозы радиации от 15 000 до 20 000 рад (поглощенная доза радиации). Для сравнения, типичная разовая терапевтическая доза радиации составляет до 200 рад. Кенстоунская клиника использовала Therac-25 с 1983 года без происшествий. Техники и фирма AECL не поверили, что эта проблема может быть вызвана Therac-25. В конечном итоге пациентка потеряла грудь, а также возможность пользоваться руками и плечами из-за радиационного поражения.

Ontario Cancer Foundation, Хемилтон, провинция Онтарио, Канада (Hamilton, Ontario, Canada)

Клиника в Хэмилтоне использовала Therac-25 в течение шести месяцев до инцидента с передозировкой. Сорокалетняя женщина поступила в клинику на 24-й сеанс лечения аппаратом Therac-25. Аппарат отключился через пять секунд после того, как оператор запустил его. Операторы были знакомы с частыми неполадками машины. Эти неполадки, вероятно, не имели серьезных последствий для пациента. Поскольку машина показывала, что облучение не было произведено, оператор попытался повторить его. Были произведены пять попыток. После пятой попытки был вызван техник, не обнаруживший проблем в аппарате.

Об инциденте сообщили в AECL, но воспроизвести неполадку и сделать заключение о причинах такого поведения Therac-25 не удалось. Однако благодаря этому сообщению фирма AECL обнаружила некоторые слабости конструкции и потенциальные механические проблемы в позиционировании поворотной платформы Therac-25 и были сделаны исправления. Пациентка умерла через пять месяцев. Результаты вскрытия показали, что смерть наступила от рака, а не от передозировки радиации. Однако вскрытие также выявило серьезные поражения бедра, вызванные радиационным воздействием. Как было позже определено, пациентка получила дозу порядка 13 000 - 17 000 рад[8].

Yakima Valley Memorial Hospital, Якима, штат Вашингтон (Yakima, Washington). Декабрь 1985

После лечения аппаратом Therac-25 у женщины развилось сильное покраснение кожи в форме параллельных полос. Персонал клиники считал, что это явление мог вызвать Therac-25. Однако они не смогли воспроизвести конфигурацию оборудования, которая была использована при лечении пациентки. Персонал проинформировал AECL о потенциальной передозировке. В AECL снова посчитали, что передозировка при использовании аппарата Therac-25 невозможна. Персонал клиники не был осведомлен о двух предыдущих случаях и не имел возможности расследовать инцидент, поэтому они не стали продолжать дело после ответа из AECL. Пациентка, вероятно, получила более низкую дозу радиации, чем два предыдущих пострадавших, и она не испытала серьезных последствий в результате передозировки[9] .

East Texas Cancer Center (Восточно-техасский онкологический центр), г. Тайлер, штат Техас (Tyler, Texas). Март 1986

В данном случае процессом управляла опытный оператор, проведшая уже более 500 подобных сеансов. Она быстро ввела предписанные параметры, после чего заметила, что вместо режима облучения электронными лучами заказала лучи рентгеновские (которыми пользовали большинство пациентов). Коррекция требовала исправления всего одной буквы; нажав кнопку, она вошла в режим редактирования, скорректировала в нужном месте "x" на "e", затем несколькими нажатиями клавиши "Return" (благо, все остальные параметры были введены правильно) достигла нижней (командной) строки экрана, убедилась, что против каждого введенного параметра горит "VERIFIED", а статус системы - ожидаемый ("BEAM READY"), и выдала команду начать процесс облучения. Однако, неожиданно система встала, на консоли высветилось сообщение "MALFUNCTION 54", а статус системы изменился на "TREATMENT PAUSE", что свидетельствовало о проблеме невысокой степени серьезности. Висевшая тут же бумага с кодами ошибок "исчерпывающе" поясняла, что "MALFUNCTION 54" означает "dose input 2". Забегая вперед, укажем, что много позже, во внутренней документации производителя было обнаружено, что это сообщение выдавалось в случае "ненадлежащей дозы облучения" - причем, как для слишком большой, так и для слишком малой, что само по себе странно (да и просто недопустимо - ведь ситуации принципиально разные). Озадаченная операторша взглянула на высветившееся количество отпущенной дозы и увидела, что оно - пренебрежимо мало. Поэтому она без долгих раздумий выдала команду на продолжение процесса, после чего вся описанная выше ситуация повторилась.

Тем временем пациент, который возлежал на столе в изолированном от оператора помещении, испытал некое подобие электрического шока. Он тоже был опытным (для него это был девятый сеанс), поэтому понял, что творится что-то неладное. Однако, дать сразу же знать об этом оператору через специально для того предназначенные видео и аудио средства он не смог: как выяснилось, видео было по непонятным причинам отключено, а аудиоканал - просто неисправен. После повторного шокового удара пациент вскочил и нимало шокировал уже операторшу, начав ломиться в стеклянные двери ее помещения. Поначалу его и лечили от электрошока (он умер через пять месяцев). Позднейшее моделирование ситуации показало, что пациент получил менее чем за 1 сек. на участок позвоночника в 1 кв. см. дозу в диапазоне от 16500 до 25000 рад (в то время, как ему было предписано принять в этом сеансе 180 рад, а всего - 6000 рад за шесть с половиной недель).

Прибывший из AECL инженер, несмотря на все усилия, оказался не в состоянии воспроизвести ситуацию, хотя заверил, что переоблучение в принципе невозможно. Были успешно прогнаны все тесты, система снова вступила в эксплуатацию

East Texas Cancer Center (Восточно-техасский онкологический центр), г. Тайлер, штат Техас (Tyler, Texas). Апрель 1986

Три недели спустя другой мужчина поступил для лечения с помощью Therac 25. Та же женщина-оператор, которая участвовала в прошлом инциденте, была ответственной за лечение. Как и в предыдущем случае, она сделала ошибку при вводе данных, и, почти столь же быстро исправив ошибку, она запустила лечение. Машина ответила сообщением «ошибка 54» и отключилась. Однако пациент уже получил передозировку, и оператор побежала за помощью. Therac-25 был отключен, и клиника проинформировала AECL о втором случае передозировки.

Физик клиники, Фриц Хэгер (Fritz Hager), совместно с оператором научились воспроизводить сообщение «ошибка 54» произвольно. Оказалось, что передозировка возникала, если данные врачебного предписания редактировались в быстром темпе. Люди из AECL наконец смогли воспроизвести ошибку и признали, что передозировка была возможна[11].

Пациент скончался через три недели после инцидента от передозировки радиации[12].

Что же было причиной?
<Enter>

В комплексе не использовалась какая-либо стандартная операционная система: была разработана специальная мультизадачная ОС реального времени, для компьютера PDP-11/23 с 32Кбайт и написанная на языке ассемблера. Специальный планировщик координировал деятельность всех одновременно исполняющихся процессов. Задачи, запускавшиеся каждые 0,1 сек., разделялись на "критические", исполнявшиеся первыми, и "некритические". К критическим отнесены три приоритетных задачи (рис. 1):

- **"Servo"**, ответственная за все операции, связанные с эмиссией радиационных пучков и доставкой их к месту назначения;
- **"Housekeeper"**, выполнявшая верификацию всех параметров и ответственная за блокировку работы в случае возникновения нештатной ситуации, а также за сообщения о таких ситуациях;
- **"Treat"**, управлявшая самим процессом лечения, который был разделен на 8 операционных фаз. В зависимости от значения переменной Tphase вызывалась одна из восьми подпрограмм, по окончании работы которой Treat - в зависимости от значений нескольких разделяемых с другими критическими и некритическими задачами переменных, вырабатывала план на новый цикл.

Одна из вызываемых Treat подпрограмм - Datent (Data entry) через разделяемую "флаговую" переменную Data_entry_complete взаимодействует с "некритической" задачей Keyboard Handler, которая управляет вводом информации с клавиатуры, исполняясь параллельно с Treat. Keyboard Handler распознает момент окончания ввода и сигнализирует об этом, изменяя значение Data_entry_complete. В свою очередь, Datent проверяет значение этой переменной. Если оно не изменилось, то значение Tphase остается равным "1", и на следующем цикле Treat опять запустит Datent; если же значение Data_entry_complete изменилось, то Datent меняет значение Tphase с "1" на "3"; в результате после окончания работы Datent монитор Treat вызовет подпрограмму Set Up Test, выполняющую проверку считающихся уже установленными параметров.

Необходимо упомянуть еще одну переменную MEOS (Mode/Energy Offset), разделяемую между Datent, Keyboard Handler и еще одной некритической задачей Hand. Старшие байты MEOS используются подпрограммой Datent для установки одного из двух режимов облучения и величины энергии испускаемого потока, в то время как младшие используются параллельно работающей задачей Hand для установки коллиматора в положение, соответствующее выбранному режиму и энергии.

Оператор мог после ввода параметров режима и энергии редактировать эти величины по отдельности. Однако, здесь присутствовал тонкий момент - разработчики установили: об окончании процесса ввода (и редактирования!) параметров свидетельствует то, что все параметры заданы и курсор находится в командной строке, на предмет чего каждые 8 сек. (величина выбрана, исходя из некоторых технических соображений, связанных с инерционностью приборов) производится опрос переменной Data_entry_complete. Если в пределах этих 8 сек. курсор покидает командную строку и - после быстрого редактирования параметров - успевает вернуться на нее, то Keyboard Handler этого события просто не заметит, и соответственно, никак переменную Data_entry_ complete не изменит. Иными словами, потенциально существует возможность для следующей последовательности действий:

- Keyboard Handler отследил местонахождение курсора на командной строке и установил флаг Data_entry_complete;
- затем оператор изменил данные в MEOS;
- не заметив этого (если к моменту опроса курсор оказался вновь на командной строке), Keyboard Handler не переустанавливает флаг Data_entry_complete;
- тогда Datent уже не способна обнаружить изменение MEOS - она свою работу заканчивает, установив Tphase=3 (а не Tphase=1, чтобы отработать еще один цикл и учесть изменения);
- тем временем, параллельно работающая Hand устанавливает коллиматор в положение, соответствующее младшим байтам MEOS (их установила ранее Datent), которые могли находиться в противоречии со старшими байтами этой разделяемой переменной (как раз и подвергшимся редактированию!). Специальных проверок для обнаружения такой несовместимости предусмотрено не было.

Сноровистая и уже набившая на этой работе руку операторша, в отличие от неторопливых инженеров AECL, скорректировала "режим" и вернула курсор обратно на командную строку очень быстро - уложившись в 8 сек. В итоге, проделанное ею изменение режима воспринято не было - он остался прежним (рентгеновским), а вот задаваемые параметры (включая находящиеся в младших байтах MEOS, критически влияющие на величину и направление потоков частиц) соответствовали электронному (фотонному) режиму. Последний штрих в катастрофическую картину внесли показания дозиметра, дававшего показания в "условных единицах" - то, что высвеченная "малая" величина дозы относилась к другому режиму и потому не подлежала рациональной оценке, операторше не пришло в голову.

Yakima Valley Memorial Hospital, г. Якима, штат Вашингтон (Yakima, Washington). Январь 1987

К этому времени проблемы с машиной Therac-25 были преданы широкой гласности, по крайней мере в сообществе пользователей. Операторы знали о запрете редактировать данные. В то время когда AECL совместно с Управлением по контролю за продуктами и лекарствами (FDA) активно занималась планом работ по устранению неполадок, случился шестой инцидент. В данном случае пациент должен был получить три дозы облучения.

Первые две дозы составляли 4 и 3 рад. Следующая доза составляла 79 рад в фотонном режиме. Первые два сеанса прошли без осложнений. После второй дозы оператор вошел в комнату облучения, чтобы повернуть поворотную платформу, чтобы проверить положение лучевого пучка относительно тела пациента. Оператор нажал кнопку возле поворотной платформы для указания того, что контроль произведен. Установив платформу, оператор запустил процесс лечения. Машина остановилась через 5-6 секунд, и поэтому оператор запустил лечение снова. И снова машина отключилась и отобразила «маловразумительную» причину остановки. Возникло предположение о передозировке, однако дисплей показал только облучение в 7 рад от двух первых сеансов.

Пациент умер в апреле 1987 году от осложнений, вызванных передозировкой. Машина была отозвана вскоре после этого[13].

Что же произошло? Выявленная в итоге расследования проблема выходит далеко за пределы частного случая еще одной программистской ошибки. В данном случае не сработала блокировка, реализованная программно позволившая прибору действовать (испускать поток фотонов) при ошибочной установке параметров. Ситуация возникла в момент, когда введенные параметры уже верифицированы подпрограммой Datent и монитор Treat в соответствии со значением переменной Tphase = 3 вызвал подпрограмму Set Up Test.

Во время установки и подгонки параметров подпрограмма Set Up Test вызывается несколько сотен раз - пока все параметры не будут установлены и верифицированы, о чем эта подпрограмма судит по нулевому значению разделяемой переменной F$mal. Если же значение ненулевое - цикл повторяется. F$mal, в свою очередь, устанавливается подпрограммой Chkcol (Check Collimator) из критической задачи Housekeeper, проверяющей, все ли с коллиматором нормально; а вызывает Chkcol другая подпрограмма задачи Housekeeper под названием Lmtchk (analog-to-digital limit checking), и вызов этот происходит, только если значение разделяемой переменной Class3 ненулевое. А ненулевым его делает как раз сама Set Up Test, которая (пока F$mal=0) каждый раз выполняет над Class3 операцию инкремента.

Эта переменная - однобайтовая, следовательно каждый 256-й проход заставляет ее сбрасываться в ноль. А ведь этот ноль - свидетельство, что все параметры, наконец, установлены. Если повезет, что именно в этот момент оператор нажмет клавишу "set" для запуска установки коллиматора в надлежащую позицию (а он это может сделать в любой момент, так как уверен, что система позволит коллиматору начать позиционироваться, только если все параметры заданы и верифицированы), то основываясь на случайно возникшем нулевом значении Class3, подпрограмма Lmtchk уже не станет вызывать Chkcol, а значит установить ненулевое значение F$mal будет некому. Иными словами, в ситуации, когда параметры не установлены должным образом (в данном конкретном случае "челюсти" коллиматора были еще раскрыты слишком широко), программная блокировка не сработала: Set Test Up установила Tphase = 2, что позволило монитору Treat прекратить цикл вызова Set Up Test, а инициализировать подпрограмму Set Up Done, по существу запускающую процесс излучения, который и потек бурным потоком, а не узеньким ручейком, как предполагалось.

Slide 15: Что делать надо (и не надо)

Не переусложняйте систему
Вспомним структуру софта Therac-25 - напоминает абстрактную картину, не так ли? Возникает вполне законный вопрос - зачем было в сугубо последовательной задаче делать десяток параллельных процессов, да еще накручивать на это свою собственную ОС? Все ошибки Therac-25 скорее всего не появились бы вообще, если бы структура системы была бы сделана простой и последовательной.

Не забывайте про синхронизацию
Если уж вам понадобилось сделать многозадачную систему, то в первую очередь ознакомьтесь с методами синхронизации процессов и совместного доступа к разделяемым ресурсам. Ошибки в этой области очень трудно поймать и практически невозможно надежно повторить.
Взаимодействие процессов и ниток (thread'ов) с ОС само по себе может быть большой тайной, даже для тех, кто на этом всем собаку съел.
Например - в небезызвестном пакете Delphi есть поддержка нитей в создаваемых GUI приложениях. Даже есть специальный класс для этого. Однако, из этих самых нитей нельзя напрямую работать с элементами экранного представления - это приводит к трудноуловимым глюкам. В вышеупомянутом классе есть специальный метод для синхронизации GUI с параллельно работающими нитями. И несмотря на это толпы программистов (и не только начинающих) с не иссекающим упорством годами ходят по одним и тем же граблям.
Другой пример - в Linux'е есть процессы, порождаемые fork'ом, и нити, порождаемые pthread_create. При этом взаимодействие между нитями и процессами весьма оригинальное - если процесс, в котором есть много нитей, выполняет fork, то дублируется только одна нить - та, которая сделала fork. Остальные молча умирают. Если остальные нити держали заблокированными какие либо примитивы синхронизации, то в новом процессе они останутся заблокированными навеки. Несмотря на то, что такое поведение задокументировано, далеко не все программисты (даже опытные) знают об этом.

Используйте те данные, которые контролировали
Если у вас есть процедуры контроля данных, то обеспечьте, что бы в дальнейшую работу шли именно те данные, которые контролировались этой процедурой. Если данные на контроль и для дальнейшей работы поступают независимо, пусть даже и из одного источника, то результатом может стать еще один Therac-25

Используйте аппаратный контроль
Если ваша система работает в таких областях, где любое неправильное поведение может стать источником катастрофы любого масштаба, то в такой системе обязательно должны быть предусмотрены многоуровневые блокировки и проверки, вплоть до аппаратных. Ошибки, обнаруженные в Therac-25 были унаследованы им от предшественника - Threrac-20. Но там они не приводили к таким последствиям, т.к. в нем были аппаратные блокировки, которые не позволяли проявится фатальным последствиям. В Therac-25 все аппаратные блокировки были убраны. AECL утверждала, что в Therac-25 абсолютно надежное программное обеспечение, с многочисленными программными блокировками, так что необходимость в аппаратных блокировках отпала.

Используйте активные системы контроля
Иногда даже аппаратные блокировки могут не справится с последствиями ошибок в системах управления. В таком случае напрашивается создание параллельно работающих

систем контроля, которые могли бы анализировать работу системы, и вмешаться, пока не поздно.

Вот примеры

Slide 16: Где нужен активный аппаратный контроль

Погнутая ось антенны
На одном из военных объектов разрабатывали программу для управления вращением параболической антенны радарной установки. Программист был гражданским, представителем производителя ЭВМ. Программа выдала команду на увеличение скорости вращения двигателя и повисла. Двигатель честно вращал антенну с все возрастающей скоростью, пока в конце концов под действием центробежной силы и разбаланса самой антенны не погнулась ось на которой она была закреплена.
Разработчика в зашей выгнали с объекта, а остаток 'творческого коллектива' вынес ему неофициальную благодарность за 2х недельный незапланированный отпуск, пока меняли ось и антенну.

Сход с рельсов робота-манипулятора
В другой (на этот раз вполне гражданской) лаборатории отрабатывали систему управления роботом-манипулятором. Вот таким <Enter>.
Робот был установлен на рельс и передвигался вдоль стены лаборатории (довольно длинной стены). Весил робот тоже прилично, так что мотор, который его двигал, был весьма и весьма мощным.
И вот в один прекрасный день система управления включила мотор на полные обороты и повисла (очень похоже на предыдущий случай, не находите?)
Ну и робот резво поехал от одной стены к противоположной. Очень и очень резво. И чем дальше, тем резвей.
На счастье, тут были предусмотрены системы блокировки - недалеко от конца рельса стоял микро выключатель, который обесточивал мотор. Кроме того, в конце рельса был механический стопор.
Увы, это не помогло. К моменту, когда робот пролетал мимо микро переключателя, у него была такая скорость, что обесточивания мотора он даже не заметил. Стопор на конце рельса был замечен, но в данном случае это скорее стопор заметил робота, который снес этот бедный стопор в момент и продолжил свой полет.
Это кстати действительно был полет, ибо рельс тоже закончился, и робот, аки птица, воспарил в небо. И приземлился в соседней лаборатории, пробив по дороге стену.
В соседней лаборатории сначала обалдело смотрели на счастье, привалившее им от соседей, а потом сказали, что у них сегодня коллективный 2й день рождения, так как никого не придавило только чудом.
В данном случае активная система безопасности должна была контролировать положение робота и ускорение, и не просто отключать мотор, а включать тормоз, если в текущем положении робот не успеет остановиться до конца рельса.

Теперь вернемся еще раз к медицинской аппаратуре, на этот раз немного другой

Slide 17: Multidata Systems

National Cancer Institute Панамы использовал установку радиационной терапии Cobalt-60. Как 8 пациентов умерло, получили передозировку как минимум 20.

В отличии от Therac-25 установка Cobalt-60 была старой, сильно поношенной и недостаточно обслуживаемой. ПО, которое работало на ней, было выпущено фирмой Multidata Systems, т.к. больгица не могла себе позволить поддержку со стороны производителя аппарата - Theratronics.

2 техника, которые работали на установке, уволились, оставив остальных работать по 16 часов в сутки, что бы обеспечивать лечение. Очень больные пациенты иногда ждали по 4-6 часов в день, что бы получить запланированное лечение.

Перегруженные работой и уставшие техники требовали некоторой софтверной поддержки, но менеджеры игнорировали их запросы.

ПО допускало применение специальных металлических щитов (они назывались 'блоки'), которые можно было положить на пациента для защиты определенных частей тела от радиации. ПО допускало до 4х таких щитов, но доктора обычно использовали 5. Что бы обойти ограничение в программе, они представляли все 5 щитов как один большой щит с отверстием в середине.

К сожалению ПО от Multidata Systems имело ошибку, которая в зависимости от того, где оказывалось отверстие в щите, увеличивало дозу в 2 раза.

Из за сильной загруженности персонала и отсутствия должного надзора со стороны менеджмента, процесс продолжался 7 месяцев.

Multidata Systems наконец признала ошибку в своем ПО и выпустила совет по поводу того, как правильно располагать дырки в щите, в середине 2001г, но к тому времени для многих пациентов было уже слишком поздно.

Но этот институт был не единственным, пострадавшим от установки Cobalt-60.

Instituto Oncologico Nacional (ION), национальный госпиталь по лечению рака, расположенный в старом комплексе Gorgas в Панаме. Они так же использовали машину от Theratron, для которой у них не было даже инструкции по эксплуатации. Доза облучения отмерялась 'на глаз' и не контролировалась. Как минимум 28 пациентов получили передозировку, некоторые из них - смертельную. К нескольким докторам были приняты дисциплинарные меры, и хотя эксплуатация такой машины без описания должно бы было стать серьезным административным проступком, оно таковым не стало. Вместо инструкции по эксплуатации у директора госпиталя (Juan Carlos Barés) был брат (Carlos Barés) - глава национальной полиции, и шурин (Ilka de Barés) - глава иммиграционной службы. Так что по правилам административного кумовства никаких вопросов к госпиталю возникнуть не могло в принципе.

Но вернемся к Multidata. Согласно докладу FDA (US Food and Drug Administration) от 7 Мая 2003 года Multidata имела долгую историю взаимоотношений с FDA.
В своем пресс-релизе FDA писало:
“The Food and Drug Administration (FDA) сегодня анонсировало, что оно запретила Multidata Systems International Corporation производить и распространять медицинские

устройства для радиационной терапии.

“Multidata не может обеспечить надлежащего уровня производства и не придерживается стандартов разработки, а так же не реагирует на требования от FDA, когда ее ставят в известность, что ее продукция могла стать причиной (или способствовала) смерти или серьезным увечьям.

"’Multidata Systems имеет 9ти летнюю историю нарушений и неспособности исправить их’ заявил уполномоченный FDA Mark B. McClellan, M.D., Ph.D. ‘Несмотря на повторяющиеся предупреждения, компания продолжает производить ее медицинское оборудование таким образом, что это подвергает риску пациентов. Это не допустимо.’”

Некоторые ошибки могут серьезно изменить степень своей серьезности в результате действия людей. Изначально незначительные, в результате ошибок в менеджменте или неправильной реакцией в средствах массовой информации они могут вырасти до гигантских размеров.

Slide 18: Ошибка в процессоре Intel Pentium

Профессор Томас Найсели - математик. В период 1993-1994 годов он работал над исследовательским проектом в области, называемой вычислительной теорией чисел (computational number theory). Одной из его целей было продемонстрировать полезность настольных персональных компьютеров. Для своего исследования он заставил большое число персональных компьютеров вычислять простые числа, пары простых чисел, триплеты и квадраплеты простых чисел для всех положительных целых до 6×10^{12}.

При вычислениях его программа предпринимала некоторое число проверок, в которых уже известные, опубликованные в литературе числа просчитывались, чтобы проверить правильность вычислений. Машина с процессором Pentium приняла участие в вычислениях в марте 1994, а 13 июня 1994 года проверка закончилась неожиданным значением. После четырех месяцев поиска профессор Найсели смог определить ошибку в блоке вычисления чисел с плавающей запятой (floating-point unit, FPU) процессора Pentium. В публичном сообщении, сделанном 9 декабря 1994 года, он описал свои трудности и исследования причины ошибок[16].

24 октября 1994 года, после того как профессор окончательно уверился в результатах анализа, он послал в службу технической поддержки Intel сообщение об ошибке. Когда Intel не ответила на сообщение, 30 октября 1994 года профессор Найсели написал письмо некоторым своим коллегам, объявив об открытии ошибки деления чисел с плавающей запятой в процессоре Intel Pentium.

Если коротко, FPU процессора Pentium возвращает ошибочное значение для некоторых операций деления. Например,

    1/824633702441,0

вычисляется некорректно (ошибочны все цифры после восьмой значащей цифры)[17].

Вскоре сообщение об ошибке уже циркулировало на форуме CompuServe и в группах новостей Интернета. Александр Вулф (Alexander Wolf), репортер EE Times, подхватил эту историю и написал статью, которая появилась в номере EE Times от 7 ноября 1994 года. Отвечая на запрос репортера, Intel заявила, что они обнаружили эту ошибку летом 1994 года, и она была исправлена в процессорах, выпущенных позднее. Однако Intel не смогла определить число выпущенных дефектных процессоров, и они попытались сгладить важность этой ошибки.

Смит (Smith), представитель Intel, подчеркнул, что этот дефект не повлияет на среднего пользователя. Говоря о Найсели, Смит сказал: «Это исключительный пользователь. Он круглосуточно проводит вычисления обратных величин. То, что он обнаружил после многих месяцев вычислений, является примером того, что восемь десятичных чисел правильны и только девятая отображается неверно. То есть ошибка у вас будет только в девятом знаке справа от точки. Я думаю, что, даже если вы инженер, вы этого не заметите».[18]

CNN распространила это заявление 22 ноября 1994 года, и вскоре оно было во всех главных средствах массовой информации, таких, как New York Times и Associated Press. В других интервью Intel повторяла свое раннее заявление о том, что ошибка несущественна для среднего пользователя.

В среду Intel заявила, что они не считают необходимым отзывать процессор, утверждая, что обычный пользователь имеет только один шанс из девяти миллиардов получить неверный результат из-за этой ошибки и таким образом не будет никаких заметных последствий для компьютеров дома и в офисе. Компания заявляет, что она продолжает поставлять сборщикам компьютеров процессоры Pentium, сделанные до того, как проблема была обнаружена[19] .

28 ноября 1994 года Тим Коу (Tim Coe) из компании Vitess Semiconductor опубликовал статью в группе новостей comp.sys.intel, в которой он путем анализа восстановил реализацию алгоритма и предложил модель поведения процессора Pentium. Через несколько дней появились аппаратные и программные «заплатки» для ошибки. 3 декабря 1994 года Воэн Пратт (Vaughan R. Pratt) из Стэндфордского университета опубликовал письмо в группах новостей comp.arch и comp.sys.intel, в котором оспаривал точку зрения Intel о том, что вероятность встречи с ошибкой составляет «один раз в 27 000 лет». Он смог продемонстрировать возможность активации ошибки один раз в каждые 3 миллисекунды в достаточно правдоподобном сценарии. А также он продемонстрировал, что достаточно безобидно выглядящее деление 4,999999/14,999999 приводило к отклонению от правильного результата на 0,00000407 при использовании дефектного процессора[20].

12 декабря 1994 года фирма IBM выпустила сообщение, в котором также был подвергнут сомнению анализ Intel о том, что вероятность обнаружения ошибки составляет один к девяти миллионам[21].

14 декабря 1994 года Intel опубликовала официальное сообщение, которое было датировано 30 ноября 1994 года[22].

В этом сообщении рассматривалась данная ошибка, обсуждались ее последствия, и этот документ был, очевидно, источником многих заявлений Intel. В этом отчете определялось, что вероятность встречи с ошибкой составляет «один к девяти миллиардам» и что среднее время появления ошибки - «один раз в 27 000 лет». Далее Intel описывала причину ошибки и алгоритм работы FPU. Инженеры компании избрали в качестве алгоритма деления процессора SRT алгоритм[23] по основанию 4 (radix 4 SRT algorithm), чтобы скорость деления могла быть удвоена, по сравнению с 486-м процессором. Полное описание SRT алгоритма, использованного в процессоре Pentium, находится за рамками этой книги. Детали можно найти в работе Edelman, Sharangpani и Bryant[24].

Однако главная причина ошибки в том, что SRT-алгоритм требует справочную таблицу для определения частного. Значение в справочной таблице генерируется численно и загружается в программируемый справочный массив. Дефект скрипта привел к тому, что несколько элементов (всего 1066, было загружено 1061) в справочной таблице были пропущены. Когда выполнялась операция деления, которая требовала эти значения, извлекалось неверное число, и точность вычисленного результата частного оказывалась сниженной.

В конечном счете 20 декабря 1994 года фирма Intel объявила, что она начинает заменять процессоры Pentium по требованию. Это существенно снизило ажиотаж вокруг ошибки в процессоре. Однако это не остановило тех, кто анализировал эту ошибку. 19 сентября 1995 года Алан Эдельман (Alan Edelman) опубликовал отчет, в котором провел детальный анализ этой ошибки[25].

В своем отчете он определил, что было только два способа, с помощью которых можно было осуществить доступ к ошибочным данным и использовать их при вычислениях. Ошибочное значение извлекалось, только если делитель содержал шесть последовательных бит, от 5-го до 10-го, установленных в единицу. Таким образом, ошибочные табличные значения могли не извлекаться при тестах, основанных на случайной выборке значения; тест, способный выявить ошибку, должен работать лучше, чем простая случайная выборка. Он также показал, что максимальная абсолютная ошибка в данном случае не могла превышать 0,00005.

В конце концов ошибка обошлась фирме Intel в $475M

Slide 19: Как реагировать на ошибки?

Можно ли добавить обработку ошибок после завершения программы?
Часто программист испытывает соблазн считать, что его программа будет работать в идеальном мире, где ошибок не случается. Так что и обрабатывать их не надо. Когда реальный мир разбивает эти мечты, такие программисты говорят - вот сейчас программу допишу, а потом добавлю к ней обработку ошибок. Насколько это реально?

Если ваша программа допускает аварийно завершиться после обнаружения любой ошибки, то такую обработку ошибок добавить можно. Но это потребует просмотреть весь код, который был написан, и возможно дописать еще столько же.
Если же нужно как то жить дальше после ошибки, то добавить такую обработку в готовую программу (в которой это не было спланировано на этапе написания) практически нереально.

Как реагировать на ошибки?
Вот программа нашла ошибку - что делать? Простой выход - сказать пользователю 'ну не смогла' и помереть. Для программы типа 'Hello world' этот подход вполне нормальный. В конце концов - какая разница напишет программа 'Hello World' или 'Good by cruel world'. Но если ваша программа немного посложнее, то такой подход вызовет массу эмоций у конечных пользователей, и лучше, если разработчик в это время будет от них подальше.

А что делать многочисленным embedded системам, которые тут в основном и рассматриваются? Ну например процессор со встроенной системой бинарной трансляции. Что он должен сделать, если она даст сбой? Как аварийно завершаться? Выдернуть ножки и выпасть из сокета?

К сожалению общего ответа на вопрос 'что делать' не существует. Каждое конкретное применение требует своего подхода.

Как тестировать программы (coverage, TDD)
Конечно лучше, если ошибок в программах вообще не будет. Увы, это недостижимая мечта. Но стремиться к этому надо. И большим подспорьем на этом пути являются тесты. Программа должна тестироваться как можно тщательнее и в наибольшем объеме.
А как определить, достаточен объем тестирования или нет?
Пример - один программист, работающий в команде, которая создавала компилятор Фортрана, придумал оптимизацию, которая должна была дать несколько процентов производительности, правда в довольно редком случае. Он реализовал оптимизацию в компиляторе, проверил и она отправилась в очередной релиз компилятора. Через пол года в компиляторе нашли багу. Исследования показали, что она была в прологе этой самой оптимизации, и проявлялась всегда, когда должна была сработать эта оптимизация. Возникает 2 вопроса - как программист ее проверял, и нужно ли было вообще ее включать в компилятор.

Итак, что же делать? Один из обычных методов - coverage. На вашей программе запускается тест и специальная утилита проверяет, какие куски кода исполнялись и сколько раз. Это позволяет сразу увидеть, если какие то части программы вообще не работали, значит они и не проверялись.
Правда возникает вопрос - а что делать с проверкой ошибок в самой программе. С одной стороны она и не должна срабатывать, а с другой стороны ее тоже неплохо было бы проверить. Для этого можно предусмотреть в самой программу искусственную вставку ошибок (под опцией)

Еще существует очень мощная технология - Test Driven Development. Вкратце она как надо добавлять функциональность и тесты и состоит из 3 шагов:

1. Сначала пишется тест на пока отсутствующую функциональность. Тест должен упасть
2. Потом в программе пишется заглушка, имитирующая отсутствующую функциональность (в объеме, достаточном для теста). Тест должен пройти
3. Заглушка убирается и пишется сама функциональность

Corner Case ошибки

Давайте рассмотрим другой аспект автоматического тестирования - какие должны быть данные для тестирования. Очень мощный класс тестов это случайные тесты. Специальный генератор создает случайным образом тестовые данные и ответы, которые должны быть получены от тестируемой программы.

Однако, несмотря на то, что такое тестирование выявляет практически все ошибки, гарантии тут дать нельзя.

Есть другой класс тестов - так называемые corner case тесты. Они не генерируются автоматически, а пишутся вручную после внимательного изучения того, что собираемся тестировать. Эти тесты используют данные, которые в тестируемой программе обрабатываются специальным образом. Например - если программа обрабатывает все числа меньше 10 одним образом, а больше 10 - другим, то тест запускается на наборе 9,10,11

Реальный случай - компилятор, отлаженный и тщательно оттестированный (в том числе и на случайных тестах) уже должен был отправится заказчику. Общее количество тестов перевалило за сотни тысяч. На всякий случай его дали протестировать человеку, который как раз и специализировался на тестировании. Он внимательно изучил исходники компилятора и написал около сотни очень простых тестов (как раз из серии corner case). Вся сотня тестов упала.

UB в программах

Никогда, ни при каких условиях не используйте в своих программах то, что в стандарте языка помечено как Undefined Behavior. Такая конструкция может работать на одном компиляторе, и неожиданно перестать работать на другом.

Например в C оператор сдвига для величины сдвига больше чем размер сдвигаемого операнда это UB.

Например сдвиг 32х битного целого на 32 бита влево может не дать 0, как вы могли бы ожидать. И он действительно не дает - сдвигаемое число не меняется, по крайней мере на gcc на IA32

Вот еще пример

Slide 20: Пример UB

Хэш-функциея "для разброса" точек (когда точка является ключом хэш-мэпа). Старшая часть дабла при преобразовании в инт отбрасывается, остаток отлично распеделен. Т.е. так было раньше, на MSVC 6 и MSVC 2003. Переезжаем на 2008. Сразу никто ничего не заметил, ибо эффект заметен только на большом объеме данных. В общем, вдруг замечаем тормоза, которых раньше вроде бы не было: помещение точек в граф подтормаживает, когда точек много. Долго чешем репу, прежде чем догадываемся заглянуть внутрь хэш-мэпа (ну кому придет в голову ковыряться в стандартных контейнерах, которые всю жизнь работали и чего бы им теперь сломаться). И видим одну корзину... т.е. имеем вместо мэпа массив. Оказывается, умники из MS тихой сапой изменили поведение при переполнении, и теперь там какая-то константа. Долго рылась в MSDN во всяких отличиях компиляторов, но никакого упоминания об изменении этого поведения не нашла.

Slide 21: Windows Genuine Disadvantage

Представленная в 2006г программа Windows Genuine Advantage никогда не была популярной среди пользователей Microsoft. У пользователей были проблемы с нахождением этих самых преимуществ (Advantage). Она никоим образом не помогала ни надежности, ни стабильности законных пользователей Windows. Все, что она делала - это помогала Microsoft в искоренении компьютерного пиратства.

Если учесть, что в конце Августа 2007 пираты были повсюду, куда не брось взгляд, эта задача была не из простых.

В пятницу, 24 Августа, некто из группы WGA нечаянно установил на WGA сервер пре-релизную версию программы, полную багов. Группа быстро откатила установку на оттестированную версию программы, но они не проверили, что багги реально были исправлены. И они не были. Так что в течении ближайших 19 часов, вплоть до 15:00 следующего дня, сервер пометил тысячи WGA клиентов по всему миру как нелегальных пользователей.

Пользователям Windows XP было сказано, что они используют пиратскую версию ОС. Пользователи Windows Vista пострадали больше - у них отключились некоторые функции, в том числе красивые темы Aero и виртуальный RAM диск ReadyBoost.

Первый официальный ответ на жалобы не помог: недовольные патроны посоветовали попробовать еще раз перепроверить установку во вторник. Но даже после того, как сервер починили в середине субботы, клиенты Vist'ы пришлось перепроверить из Windows системы что бы вернуть себе ReadyBoost и Aero

Конечно, это было сравнительно небольшая проблема в инженерной команде, и строго говоря, это была человеческая ошибка. Но когда ошибка касается распространения не тестированного и сбойного ПО, и когда вы примите во внимание количество людей, которых это затронет, то станет ясно, что уровень вызванного гнева и эффект домино от отрицательных публикаций в прессе, сделает такую ошибку гораздо серьезнее, чем это казалось на первый взгляд.

Slide 22: Катастрофическое исследование

Цитата из журнала:

В выпуске от 5 Февраля, мы представили результаты изучения того, как природные катастрофы влияют на количество самоубийств. Предыдущие исследования среди жертв катастроф показывали увеличенную распространенность посттравматического стрессового расстройства и депрессии, что является признанным фактором риска для суицидальных настроений. В нашей статье мы привели данные по увеличению количества самоубийств среди жителей 377 штатов затронутых единственной, но серьезной природной катастрофой произошедшей между 1982 и 1989 годами.

С прискорбием сообщаем, что мы обнаружили ошибку в компьютерной программе и что наши предыдущие результаты были некорректны. Ошибка обнаружилась, когда мы попытались повторить исследование что бы определить были ли изменения в количестве самоубийств после катастрофы. Когда мы проследили процесс сбора данных о самоубийствах, мы обнаружили, что смерти в 1990г были посчитаны дважды. После того, как ошибка была устранена, новый анализ показал, что уровень самоубийств не увеличился, как для всех типов катастроф вместе, так и для каждого отдельного типа по отдельности.

Slide 23: McAfee - перезагрузка

April 21, 2010 11:56 AM PDT

McAfee's popular antivirus software failed spectacularly on Wednesday, causing tens of thousands of Windows XP computers to crash or repeatedly reboot.

A buggy update that the company released early in the day turned the software's formidable defenses against malicious software inward, prompting it to attack a vital component of Microsoft Windows. The update was available for business customers for about four hours before distribution was halted, McAfee said.

"McAfee is aware that a number of customers have incurred a false-positive error due to incorrect malware alerts on Wednesday, April 21," a McAfee statement posted on CNet.com said. "The problem occurs with the 5958 virus definition file (DAT) that was released on April 21."

The damage was widespread: the University of Michigan's medical school reported that 8,000 of its 25,000 computers crashed. Police in Lexington, Ky., resorted to hand-writing reports and turned off their patrol car terminals as a precaution. Some jails canceled visitation, and Rhode Island hospitals turned away non-trauma patients at emergency rooms and postponed some elective surgeries.

"For PCs that have been affected and are in a state of reboot, Intel IT is still working on how to get the deleted files back on the operating system, which will allow PCs to boot normally again," spokesman Bill MacKenzie told The Oregonian.

Peter Juvinall, systems administrator at Illinois State University, said that when the first computer started rebooting it quickly became evident that it was a major problem, affecting dozens of computers at the College of Business alone.

"I originally thought it was a virus," he said. When the tech support people concluded McAfee's update was to blame, they stopped further downloads of the faulty software update and started shuttling from computer to computer to get the machines working again.

In many offices, personal attention to each PC from a technician appeared to be the only way to fix the problem because the computers weren't receptive to remote software updates when stuck in the reboot cycle. That slowed the recovery.

In Utah, at least 700 of Utah Valley University's 5,000 computers on campus were affected, but university spokesman Chris Taylor said all computers were back up and running by noon Wednesday, as IT officials "were right on top of it."

In Sarasota County, Fla., school district officials said about 800 computers experienced the problem, and power was pulled quickly on the PCs. Officials said they were able to get computer systems up in running again in a little more than half an hour.

A CNET editor in Portland, Ore., was affected Wednesday morning when the update caused her computer to lose network and Internet connections and McAfee prevented her from launching programs or uninstalling it.

Tech-related mailing lists soon began buzzing with complaints. And the condemnation on Twitter was unrelenting, with Sonny Hashmi, the deputy chief information officer of the District of Columbia, calling it a "huge disruption," adding that McAfee is now on his "blacklist." An engineer in San Francisco said that, thanks to McAfee, "the wait at my work is two days and growing to get your laptop back." Others complained that, approximately six hours after the problem was known, McAfee has yet to post a note on the home page--which currently boasts of "technology to supercharge your network security."

The update released at 6 a.m. PT effectively redirected the PC's immune system, causing it to attack a legitimate operating system component known as SVCHOST.EXE in the same way that some diseases can cause the human immune system to turn inward. In this case, McAfee's application incorrectly confused it with malware known as the W32/Wecorl.a virus.

McAfee apologized to customers for the problem, which seemed to affect primarily Windows XP computers running Microsoft's Service Pack 3, but downplayed its impact. "We are not aware of significant impact on consumers," the company said in a statement sent to CNET at 2 p.m. PT.

That didn't endear the company to the enterprise users who were the most affected by the update, especially system administrators who were forced to trek from computer to computer and manually install the repair that McAfee had made available by midday. It's not clear how many customers were affected, and a McAfee representative said she did not have an update. (Here's a related CNET article on how to fix your McAfee-crippled PC.)

McAfee has posted a Web page on a separate site with detailed instructions on how to fix XP computers that have been crashing because of Wednesday's update. It recommends manually downloading and installing an "EXTRA.DAT" file and then restore files that have been incorrectly quarantined.

But that option requires a least a modest amount of technical ability, and as of 4 p.m. PDT, the company had not offered a better way. "McAfee is continuing to work on an automated solution," the page said.

But after tens of thousands of irate users flooded into the forums, the site abruptly went offline and began to return an error message.

Last update 10:20 p.m. PT: McAfee has posted a statement saying that the problem affected "one half of 1 percent of our enterprise accounts globally and a fraction of that" among home users. Another post from Barry McPherson, executive vice president for customer service, apologizes for the snafu and says: "Mistakes happen. No excuses." And it looks like the company has posted some more details about how SVCHOST.EXE was targeted.

It's not uncommon for antivirus programs to misidentify legitimate files as viruses. Last month, antivirus software from Bitdefender locked up PCs running several different versions of Windows.

However, the scale of this outage was unusual, said Mike Rothman, president of computer security firm Securosis.

"It looks to be a train wreck," Rothman said.

Slide 24: Вертолет Chihook ZD576

In 1994 in Scotland, a Chinook helicopter crashed and killed all 29 passengers. While initially the pilot was blamed for the crash, that decision was later overturned since there was evidence that a systems error had been the actual cause.

At the time of the crash, new FADEC (Full Authority Digital Engine Control) equipment was being integrated onto all RAF Chinooks, as part of an upgrade from the Chinook HC.1 standard to the newer Chinook HC.2 variant. The Ministry of Defence was given a £3 million settlement from Textron, the manufacturers of the system, after a ground-test of the FADEC systems on a Chinook in 1989 resulted in severe airframe damage. Contractors, including Textron, had agreed that FADEC had been the cause of the 1989 incident and that the system needed to be redesigned.[34]

The committee investigating the crash were satisfied that the destructive error in 1989 was not relevant to the 1994 crash.[27]

EDS-SCICON was given the task of independently evaluating the software on the Chinook HC.2 in 1993. According to the House of Commons report: "After examining only 18 per cent of the code they found 486 anomalies and stopped the review... intermittent engine failure captions were being regularly experienced by aircrew of Chinook Mk 2s and there were instances of uncommanded run up and run down of the engines and undemanded flight control movements".[6]

Tests upon the Chinooks performed by the MOD at Boscombe Down in 1994 reported the FADEC software to be "unverifiable and ... therefore unsuitable for its purpose".[6] In June 1994, the MoD test pilots at Boscombe Down had refused to fly the Chinook HC.2 until the engines, engine control systems and FADEC software had undergone revision.[19] In October 2001, Computer Weekly reported that three fellows of the Royal Aeronautical Society had said that issues with either control or FADEC systems could have led to the crash.[37]

Slide 25: К вопросу о качестве ПО

Исследования British Royal Signals and Radar Establishment
В конце 80-х гг. такая влиятельная в оборонных кругах организация как British Royal Signals and Radar Establishment сделала попытку оценки распространенности дефектов в ПО, написанном для ряда очень ответственных систем. Оказалось, что "до 10% программных модулей и отдельных функций не соответствуют спецификациям в одном или нескольких режимах работы" [7]. Такого рода отклонения были обнаружены даже в ПО, прошедшем полный цикл всестороннего тестирования. Хотя большинство обнаруженных ошибок были признаны слишком незначительными, чтобы вызвать сколь-либо серьезные последствия, все же 5% функций могли оказывать разного рода значимое негативное воздействие на поведение всей системы. Примечательно, что среди прочего авторы исследования особо упомянули выявленную в одном из модулей неназванной системы потенциальную возможность переполнения в целой арифметике, что могло привести к выдаче команды приводу повернуть некую установку не направо (как следовало), а налево. Достаточно предположить, что речь в ПО шла об управлении ориентацией пусковой ракетной установки, чтобы представить возможные последствия.

Исследования NASA по программе Shuttle
NASA инвестировала огромные средства и ресурсы в верификацию и сопровождение программного обеспечения для космических кораблей Shuttle. Несмотря на это, за 10-летие с 1980 г. - времени начала использования ПО - выявлено 16 ошибок "первой степени серьезности" (способных привести к "потере корабля и/или экипажа"). Восемь из этих ошибок не были обнаружены своевременно и присутствовали в коде во время полетов, хотя, к счастью, без последствий. Зато во время полетов были задокументированы проблемы, возникшие от проявившихся 12 значимых ошибок, из которых три относились ко "второй степени серьезности" ("препятствуют выполнению критически важных задач полета").

EDS Drops Child Support (2004)

Cost: £539 million and counting

Disaster: Business services giant EDS developed a computer system for U.K.’s Child Support Agency (CSA) that accidentally overpaid 1.9 million people, underpaid another 700,000, had £3.5 billion in uncollected child support payments, a backlog of 239,000 cases, 36,000 new cases “stuck” in the system, and still over 500 documented bugs.

Cause: EDS introduced a large, complex IT system to the CSA while trying to simultaneously restructure the agency. (more)

In 2004, EDS software giant introduced a large, complex IT system to the U.K.’s Child Support Agency (CSA). At the exact same time, the Department for Work and Pensions (DWP) decided to restructure the entire agency. The restructure and the new software were completely incompatible, and irreversible errors were introduced as a result. With over 500 bugs still reported as open in the new system, the clash of the two events has crippled the CSA’s network. *Result - The system somehow managed to overpay 1.9 million people, underpay another 700,000, had $7 billion in uncollected child support payments, a backlog of 239,000 cases, 36,000 new cases “stuck” in the system, and has cost the UK taxpayers over $1 billion to date.*

Прекращение проекта ФБР Trilogy (2005)

Cost: $105 million, still no effective case file solution

Disaster: The FBI scrapped its computer systems overhaul after four years of effort. The Virtual Case File project was a massive, integrated software system for agents to share case files and other information.

Cause: Mismanagement, and an attempt to build a long-term project on technology that was outdated before the project completed, resulted in a complex and unusable system. (more)

On Feb. 3, 2005, Robert S. Mueller III, director of the Federal Bureau of Investigation, appeared before a Senate subcommittee to explain how the FBI had managed to waste US$104.5 million.

This couldn't have been a very comfortable position to be in.

In 2001, the FBI had launched the Trilogy project, a project designed to update the FBI's IT infrastructure.

Trilogy came in three parts: update the network, update the hardware and update the software. Guess which one Robert Mueller was talking about?

That's right—the software. "Virtual Case File," or VCF, was intended to allow FBI agents to upload information to a centralized database so that it could be easily accessed by others.

It was a disaster, and the $170 million project was canceled for a loss of $104.5 million.

Sadly, the loss was entirely avoidable: The FBI made many classic mistakes. In this article, I'll take a look at three of these mistakes and apply lessons learned from the agile software movement.

Classic Mistake No. 1: Ignoring the Users.

there is no indication that the FBI made the involvement of actual agents a priority in VCF development.

User involvement isn't easy. At the March meeting of the Portland Software Development Roundtable, a participant described a government project for the state of Oregon. For several periods of six months, different groups of seven to eight users were physically moved to the capitol in order to work out detailed requirements. Larger user groups did extensive testing and review close to each release.

Ultimately, development involved 1,000 users around the state. This was no small investment, but it resulted in a successful project.

Admittedly, the users with the best understanding of what's needed are often the ones who are the busiest doing other important things.

It can be hard to get their time. It's tempting to say that you don't need those users and hope that you can get by with just your business analysts.

Got $104.5 million to burn to find out?

Classic Mistake No. 2: Expecting Requirements to Remain Stable.

Requirements always change. Always.

The FBI dealt with this fact of life in the same way many companies do: by complaining about it. This is a slightly less sophisticated form of the “change control board,” a thinly disguised mechanism for preventing requirements changes. Either way, preventing requirements changes just means that the software you deliver is less likely to do what your users need.

Classic Mistake No. 3: “Big Bang” Deployment. VCF was meant to be a replacement for a mainframe-based “green-screen” application. The FBI decided to deploy VCF with a “big bang” approach: They would develop a full replacement for the old application, switch off the old one, and switch on the new one.
This is a foolhardy approach. From a technical perspective, it means that you have to have a complete replacement for all of the functionality in the old system.

Often, it means that even defects in the old system have to be replicated, because other systems depend on those defects to work properly. It’s an immense undertaking, one that’s very difficult to get right.

Even with perfect technical execution, a big-bang deployment is foolish from an economic perspective as well. Big-bang deployment means that you can’t deploy anything until the end of the project. That means you see no benefit—no return on investment—until the very end of the project.

The FBI director’s testimony before the Senate shows that the FBI has recognized Classic Mistake No. 3 but not the other two.

Британский паспорт в никуда

Cost: £12.6 million, mass inconvenience

Disaster: The U.K. Passport Agency implemented a new Siemens computer system, which failed to issue passports on time for a half million British citizens. The Agency had to pay millions in compensation, staff overtime and umbrellas for people queuing in the rain for passports.

Cause: The Passport Agency rolled out its new computer system without adequately testing it or training its staff. At the same time, a law change required all children under 16 traveling abroad to obtain a passport, resulting in a huge spike in passport demand that overwhelmed the buggy new computer system. (more)

A new £230m computer system was brought into use at Newport Passport Office before staff were properly trained and the system tested, according to a report by the National Audit Office.

The report also said at least 500 holidaymakers across the UK missed their departure dates, following problems with a new computer system which left the agency unable to issue passports on time.

Total compensation currently amounts to £161,000, but is likely to rise further.

The problem arose because management failed to check a new Siemens computer system properly before it was introduced, and failed to make contingency plans if something went wrong, said the auditors.

David Davis, chair of the Commons Public Accounts Committee, said: "The total inability of the Passport Agency this summer to cope with the demands of its clients was a complete fiasco."

The disclosure came days after the Home Office admitted passport fees could go up from the current £21 to pay for the £12.6m cost of emergency measures.

The cost includes £6m towards staff overtime, £16,000 for umbrellas for applicants forced to queue all day in the rain for their passports and £161,000 compensation to holidaymakers, but that figure is likely to rise further.

The Passport Agency lost its Charter Mark - awarded for excellence in public service - earlier this year as a result.

Slide 26: M247 Sergeant York

M247 Sergeant York — зенитная самоходная установка, разработанная в США в начале 1980-х годов и планировавшаяся к принятию на вооружение. ЗСУ «Сержант Йорк» должны были составить основу ближней ПВО американских механизированных войск и обеспечить защиту от низколетящих самолётов, и, особенно, от вертолётов, вооружённых ПТУР.

После рассмотрения конкурсных проектов, 13 января 1978 года было отдано предпочтение предложениям «Дженерал Дайнэмикс» и «Форд Аэроспейс». Их конкурсные машины получили наименования XM246 и XM247 соответственно. Испытания начались летом 1980 года на военной базе Форт-Блисс. В ходе испытаний обе установки провели около 200 стрельб и сбили 2 радиоуправляемых самолёта, 5 вертолётов и 23 малых воздушных мишени. Результаты испытаний не давали оснований для однозначных выводов о превосходстве одной из представленных систем, скорее признавалось, что обе являются явно недоведёнными.

Тем не менее, 7 мая 1981 года победителем испытаний была признана установка производства «Форд Аэроспейс». Данное решение вызвало волну критики, но с компанией был заключён контракт на доработку и производство опытной партии из 12 единиц в 1981 году, а сама ЗСУ получила официальное название «Сержант Йорк» и индекс M247. В 1982 году предполагалась закупка 50 единиц, а всего сухопутные силы США должны были получить 618 таких ЗСУ в период с 1985 по 1989 годы на сумму около 5 миллиардов долларов.

Ford's prototype vehicle started demonstrating problems almost immediately. The main concerns had to do with the tracking radar, which demonstrated considerable problems with ground clutter. In testing, it was unable to distinguish between helicopters and trees. When the guns were pointed upward to fire on high-angle targets, the barrels projected into the radar's line of sight and further confused the system. Additionally, the reaction time was far too slow; against hovering helicopters it was 10 to 11 seconds, but against high-speed targets it was from 11 to 19, far too long to take a shot.[7][19]

In February 1982 the prototype was demonstrated for a group of US and British officers at Fort Bliss, along with members of Congress and other VIPs. When the computer was activated, it immediately started aiming the guns at the review stands, causing several minor injuries as members of the group jumped for cover. Technicians worked on the problem, and the system was restarted. This time it started shooting towards the target, but fired into the ground 300 m in front of the tank. In spite of several attempts to get it working properly, the vehicle never successfully engaged the sample targets. A Ford manager claimed that the problems were due to the vehicle being washed for the demonstration and fouling the electronics.[19] In a report on the test, Easterbrook jokingly wondered if it ever rained in central Europe.[16]

As early production examples started rolling off the production line, the problems proved to be just as serious. One of the early models is reported to have locked onto a latrine fan, mistaking its return for a moving target of low-priority. Reporting on the incident in another article on the vehicle's woes, Easterbrook reported that "During a test one DIVAD locked on to a latrine fan. Michael Duffy, a reporter for the industry publication *Defense Week*, who broke this aspect of the story, received a conference call in which Ford officials asked him to describe the target as a 'building fan' or 'exhaust fan' instead."[20]

В связи с выявленными недостатками, программа производства M247 была отменена в 1985 году.

Во время испытаний около неё безуспешно кружил вертолет-мишень, которую она так и не смогла распознать. Зато распознала как вертолет вентилятор в туалете, расположенном метрах в 800-х от ЗСУ.

И успешно его поразила.

Slide 27: Авария на телефонной компании AT&T

1990

On Jan. 15, 1990, around 60,000 AT&T long-distance customers tried to place long-distance calls as usual -- and got nothing. Behind the scenes, the company's 4ESS long-distance switches, all 114 of them, kept rebooting in sequence. AT&T assumed it was being hacked, and for nine hours, the company and law enforcement tried to work out what was happening. In the end, AT&T uncovered the culprit: an obscure fault in its new software.

В такой телефонной сети, когда один из узлов (4ESS) сталкивается с проблемой, он посылает сообщение «не беспокоить» всем узлам, с которыми он соединен. Это сообщение информирует соседний узел о том, что данный узел не может обрабатывать новые вызовы и просит соседний узел считать его не работающим. Тем временем аварийный узел активирует процесс восстановления после сбоя, который длится от четырех до шести секунд. По окончании процесса восстановления аварийный узел посылает сообщение, известное как Начальное адресное сообщение (Initial Address Message, IAM), всем соседним узлам, сообщая им о своем новом статусе и требуя направлять вызовы на восстановленный узел.

В середине декабря 1989 года AT&T произвела обновление программного обеспечения на коммутаторах 4ESS с целью увеличения производительности системы и введения быстрого процесса восстановления после ошибки. Приблизительно в 2:30 по Восточному стандартному времени (EST) 15 января 1990 года на 4ESS коммутаторе в Нью-Йорке возникла небольшая аппаратная проблема, и коммутатор начал процесс восстановления, как было описано выше. После того как Нью-Йоркский коммутатор исправил проблему, он послал сообщение IAM для уведомления соседних коммутаторов, что он готов продолжать работу. Однако обновление программ, проведенное в середине декабря, внесло в действия ошибку. Эта ошибка проявилась, когда коммутатор получил два IAM сообщения с интервалом 1/100 секунды. Некоторые данные в коммутаторе оказались искажены, и он прекратил обслуживание, перейдя к инициализации. Когда соседние узлы выходили из строя, они запускали тот же самый процесс восстановления. Поскольку все коммутаторы были одинаковы, та же последовательность событий каскадом распространялась от одного коммутатора к другому и вывела из строя всю систему.

В течение дня инженеры AT&T смогли стабилизировать сеть, уменьшив нагрузку на нее. К 23:30 EST они смогли очистить все звенья сети, и система практически вернулась к нормальному состоянию.

Во вторник, 16 января 1990 года, инженеры AT&T смогли идентифицировать и выделить ошибку, которая была отслежена до набора ошибочных кодов. Этот код активировался в ходе процедуры восстановления коммутатора. Ошибка была абсолютно идиотская - программист использовал оператор break для выхода из тела оператора if (как он думал). На самом деле break вышел из ближайшего switch'a

Вывод: авария 1990 года

Программная ошибка, приведшая к аварии 1990 года, - это типичная ошибка новичка. Но ошибки делаем все мы, и даже ветеран с 20-летним стажем может совершить случайно глупую ошибку. Если ошибки неизбежны, вопрос состоит в том, что мы можем сделать для того, чтобы отыскать ошибку до того, как она станет достоянием общества? У нас нет никакого знания из первых рук о внутренней кухне разработки программ в AT&T в 1990

году. Следовательно, мы не можем оценить вклад других причин. В официальном отчете AT&T говорилось:
Мы считаем, что процессы планирования, разработки и тестирования программ, которые мы используем, базируются на прочных и качественных основах. Все будущие программы будут также скрупулезно тестироваться. Мы используем опыт, приобретенный при решении этой проблемы, для дальнейшего улучшения наших методов[42].

Мы не считаем возможным обвинять в аварии 1990 года процесс разработки программного обеспечения в AT&T и не имеем оснований считать, что AT&T не протестировала тщательно обновление для своих программ. Оглядываясь назад, легко говорить, что, если бы разработчики только протестировали свои программы, они сразу увидели бы эту ошибку. Или то, что, если бы они проверили код, они, возможно, обнаружили бы дефект. Проверка кода может быть и помогла бы в данном случае. Однако единственный случай, когда проверка кода помогла бы найти данную ошибку, если бы другой специалист увидел именно эту строку кода и спросил первого программиста, входил ли этот код в его (ее) намерения. А единственная причина, по которой этот специалист мог бы задать такой вопрос, - знакомство со спецификацией к этому конкретному блоку кода.

Ошибки такого рода обычно нелегко выявить при тестировании в обычной тестовой среде. Такую ошибку можно воспроизвести, как только вы поймете ее и создадите последовательность действий для ее активации. Однако шанс создать правильную последовательность событий случайным образом очень мал, особенно если система будет использоваться в крупномасштабной среде реального времени, которую трудно имитировать в лабораторных условиях. Более того, новое программное обеспечение работало правильно примерно в течение месяца, что соответствует нескольким миллиардам обработанных вызовов. В программном обеспечении был дефект, но этот дефект требовал целого набора специфических событий и факторов, для того чтобы пробудиться к жизни.

- Эта ошибка требовала, чтобы нагрузка на сеть была продолжительной. Когда нагрузка на сеть уменьшалась, действие дефекта, по существу, исчезало[43].
- Эта ошибка зависела от временных параметров. Чтобы ошибка активировалась, было необходимо, чтобы были получены два IAM сообщения от одного и того же коммутатора с интервалом менее 10 миллисекунд.
- Тот факт, что на всех коммутаторах было установлено одинаковое программное обеспечение, делал систему расширяемой. Однако был риск в том, что, если на всех коммутаторах был один и тот же дефект, они все оказывались чувствительными к одной и той же ошибке.

1998

AT&T's frame-relay data network crashed Monday (April 13, 1998) shortly before noon Pacific time. By yesterday (April 14) morning, the company said it was 96 percent restored, but work continued well into the day. Voice communications, private data lines and standard Internet services weren't disrupted.

The network collapse put more than 1,000 Wells Fargo automatic teller machines out of commission and left 600 branch offices out of touch with the bank's central computers in Roseville, said Wells Fargo spokeswoman Lisa Rossi. Customers still could deposit or withdraw money from tellers, but their transactions couldn't be recorded until early yesterday morning.

Dozens of employees of Roseville Telephone Co., MCI and 3Com Corp., a major network-equipment supplier, worked furiously to set up alternative data lines between Wells Fargo's data center and all the affected branches and ATM machines. By 4 a.m., all the backup lines were in place. "It was pretty amazing to see the turnaround time," Rossi said. Merchants whose banks used AT&T's frame-relay network to communicate with

Visa in some cases couldn't make credit card sales, said Visa spokesman Da vid Sandor. "There were isolated delays in processing sales or no sales at all," he said. Most Visa transactions managed to go through on backup networks, however.

But American Express travel offices across much of the country were stymied for nearly 24 hours, said a customer service representative in Dearborn, Mich. "Agents could still call the airlines, but it drastically cut down our ability to issue tickets," she said. "Even our hotline was wiped out. We are backed up with calls galore."

13 апреля 1998 года в 2:30 после полудня к коммутатору системы ретрансляции кадров Cisco Stratacom BPX был направлен техник для обновления транк-карты (trunk-card)[44]. Коммутатор Stratacom BPX содержал две транк-карты, одна из которых была активной, тогда как другая находилась в ждущем режиме и выполняла функцию резерва. Фирма AT&T использовала две процедуры обновления транк-карт. Одна процедура использовалась, если коммутатор был в текущее время подключен к сети и активен, тогда как другая процедура применялась, если коммутатор был изолирован, то есть не соединен с сетью.

Согласно первому сценарию, то есть когда коммутатор считался активным, процедура требовала, чтобы техник заменил сначала карту, находящуюся в ждущем режиме. Как только становилось ясно, что состояние новой карты стабильно, старая активная карта переводилась в ждущий режим, а новая карта становилась активной. Проведя эту операцию, техник мог заменить оставшуюся карту (теперь находящуюся в ждущем режиме). При второй процедуре предполагалось, что коммутатор отключен от сети и техник мог менять обе карты одновременно.

Когда техник прибыл на место, он посчитал, что коммутатор, которому требуется обновление, не подключен к сети, поскольку казалось, что через него не проходил никакой сетевой трафик. Однако коммутатор был подключен к сети и активен. К несчастью для техника и для AT&T, обе карты имели дефекты. Как только карты были установлены и активированы, они немедленно выслали коммутатору поток сообщений об ошибках. Эти сообщения от транк-карт активировали ошибку в программном модуле коммутатора. Этот дефект вызвал распространение передачи сообщений об ошибках к другим коммутаторам сети, ко всем 145. Объем этих посланий был достаточно велик, для того чтобы быстро перегрузить все коммутаторы, что очень действенно вывело из строя всю систему приблизительно к 3:00 пополудни[45].

Информации об ошибках в программном обеспечении транк-карт и коммутатора Cisco немного. Элка Ярвис (Alka Jarvis), менеджер по программному обеспечению Cisco Systems, 28 мая 1998 года на заседании сессии Международной недели качества программного обеспечения (International Software Quality Week) прокомментировал, что код, который вызвал аварию в сети AT&T, являлся наследством прошлого[46].

Компания AT&T смогла быстро изолировать аварийный коммутатор, к 23:00 он был отключен от сети. Оставшаяся задача состояла в том, чтобы просто перестроить всю сеть, одну часть за другой. К 2:00 пополудни 14 апреля 1998 года 99,9 % сети ретрансляции

кадров были снова работоспособны. Однако определение причины аварии заняло у AT&T около недели, и 22 апреля 1998 года фирма выпустила отчет, очерчивающий причину выхода сети из строя.

Вывод: авария 1998 года

Хотя в аварии 1998 года и участвовало программное обеспечение, существует множество причин, внесших свой вклад в данное происшествие. Эта авария отличалась от аварии 1990 года тем, что процедурная ошибка в ходе обновления запустила скрытые программные дефекты. Однако сходств очень много.

- Установка нового программного обеспечения запустила ошибку. Программы, и старые и новые, имели многочисленные скрытые дефекты, которые не были обнаружены в ходе обычных тестовых процедур. Наличие скрытых дефектов изменило функциональную среду и запустило дефект, вызвавший аварию.
- Ошибочный код не был проверен должным образом. При аварии 1990 года это был новый код от AT&T. При аварии 1998 года - старый код от Cisco Systems.
- Программные дефекты в обоих случаях представляли собой проблемы со скрытыми граничными условиями, которые было трудно протестировать и которые, по всей вероятности, так и не были протестированы.

Авария 1998 года в сети AT&T выявила многочисленные просчеты в процедурах и процессах обслуживания сети и продемонстрировала трудности в разработке и поддержании в рабочем состоянии надежной сети. Очевидно, AT&T извлекла урок из своих ошибок и исправила многочисленные процедурные и функциональные недостатки и ввела многочисленные планы для восстановления после аварий, чтобы минимизировать риск другой общесистемной аварии[47].

Slide 28: Ракетный крейсер *Yorktown*

From 1996 *Yorktown* was used as the testbed for the Navy's Smart Ship program. The ship was equipped with a network of 27 dual 200 MHz Pentium Pro-based machines running Windows NT 4.0 communicating over fiber-optic cable with a Pentium Pro-based server. This network was responsible for running the integrated control center on the bridge, monitoring condition assessment, damage control, machinery control and fuel control, monitoring the engines and navigating the ship. This system was predicted to save $2.8 million per year by reducing the ship's complement by 10%.

On 21 September 1997, while on maneuvers off the coast of Cape Charles, Virginia, a crew member entered a zero into a database field causing a divide by zero error in the ship's Remote Data Base Manager which brought down all the machines on the network, causing the ship's propulsion system to fail.[5]

Anthony DiGiorgio, a civilian contractor with a 26-year history of working on Navy control systems, reported in 1998 that *Yorktown* had to be towed back to Norfolk Naval Station. Ron Redman, a deputy technical director with the Aegis Program Executive Office, backed up this claim, suggesting that such system failures had required *Yorktown* to be towed back to port several times.[6]

Wired 24 July 1998 - Sunk by Windows NT

In 3 August 1998 issue of *Government Computer News*, a retraction by DiGiorgio was published. He claims the reporter altered his statements, and insists that he did not claim the *Yorktown* was towed into Norfolk. *GCN* stands by its story.[7]

Atlantic Fleet officials also denied the towing, reporting that *Yorktown* was "dead in the water" for just 2 hours and 45 minutes.[6] Captain Richard Rushton, commanding officer of *Yorktown* at the time of the incident, also denied that the ship had to be towed back to port, stating that the ship returned under its own power.[8]

Atlantic Fleet officials acknowledged that the Yorktown experienced what they termed "an engineering local area network casualty."[6] "We are putting equipment in the engine room that we cannot maintain and, when it fails, results in a critical failure," DiGiorgio said.[6]

Even though the problem was caused by programming error in the Remote Data Base Manager application and not by problems with the operating system itself, criticism of operating system choice ensued. Ron Redman, deputy technical director of the Fleet Introduction Division of the Aegis Program Executive Office, said there have been numerous software failures associated with NT aboard the Yorktown.[6]

> “Because of politics, some things are being forced on us that without political pressure we might not do, like Windows NT. If it were up to me I probably would not have used Windows NT in this particular application ... Refining that is an ongoing process ... Unix is a better system for control of equipment and machinery, whereas NT is a better system for the transfer of information and data. NT has never been fully refined and there are times when we have had shutdowns that resulted from NT."”
>
> —Ron Redman[6]

LETTERS TO THE EDITOR

- Aug 03, 1998

DiGiorgio denies reported statements

Regarding your story on the USS Yorktown [GCN, July 13, Page 1], just for the record:

I did not say that the Yorktown was towed into Norfolk.

Logic does not allow the combination of the words “self-proclaimed whistle-blower.” It may sound good, but there is no such thing.

Your reporter seems to have spent his formative years as a reporter at the National Enquirer. This is based on [his] disregard of what was actually said for the sake of writing what he wanted to write.

Thanks for introducing me to being quoted. The way you obtained the information and the way you misused it is beginning to make me think that, maybe, there is validity to the claim made by the average American redneck that the media is the problem.

Anthony DiGiorgio
Engineer
Atlantic Fleet Technical Support Center
Norfolk, Va.

Editor’s note: GCN stands by its story.

Smart Ship inquiry a go

- By Gregory Slabodkin
- Aug 31, 1998

Navy chief information officer Ann Miller is conducting a detailed inquiry of the incident. The Yorktown is the Navy’s test bed for its Smart Ship program, which seeks to reduce crew workloads and operating costs by using shipboard PC systems running under Microsoft Windows NT.

The Navy CIO Office is trying to determine whether the crash was caused by the software application, NT or some other problem.

“So far, it doesn’t seem like it’s an NT issue but a basic programming problem,” said deputy CIO Ron Turner, who is in charge of the inquiry.

Microsoft officials strongly deny that NT caused the Yorktown’s systems to fail. The responsibility for ensuring ship operations doesn’t rest with the OS but with Yorktown’s system administrators and software programmers, who should have safeguarded the application from propagating the errors, company officials said.

The Yorktown's Standard Monitoring Control System administrator entered zero in the data field for the Remote Database Manager program, causing the buffer overflow, Navy officials said. **Administrators are now aware of the problem of entering zero in the database and are trained to bypass a bad data field and change the value if such a problem occurs again, Navy officials said.**

Between July 1995 and June 1997, the Yorktown lost propulsion power to buffer overflows twice while using the new Smart Ship technology, said Capt. Richard Rushton, commanding officer of the Yorktown at the time of the failures. But in each incidence the Yorktown crew knew what caused the failure and quickly restored systems, Rushton said.

Because the ships' new propulsion control system was developed quickly, his programmers knew there were inherent risks, Rushton said.

"We pushed the envelope and knew that events such as what happened in September of last year were possible," he said.

The Yorktown is equipped with two FFG-7 emergency power units in the event of a propulsion system failure, he said.

"NT was never the cause of any problem on the ship," Rushton said. "The problems were all in programs, database and code within the individual pieces of software that we were using."

But some Navy officials are concerned that NT does not have the capability to protect the network from crashing when applications fail.

"Using Windows NT, which is known to have some failure modes, on a warship is similar to hoping that luck will be in our favor," wrote Anthony DiGiorgio, an engineer with the Atlantic Fleet Technical Support Center, in a June 1998 article titled "The Smart Ship is Not The Answer."

The article appeared in the U.S. Naval Institute's Proceedings magazine and is posted on the Web at http://www.usni.org/Proceedings/digiorgio.htm.

"It took two days of pierside maintenance to resolve the [Yorktown] problem, and there have been similar failures in the past when the ship has had to be towed into port," DiGiorgio noted.

Rushton denied that the Yorktown ever had to be towed into port; it returned to port using emergency power in the September incident, he said.

"The Yorktown should not be held to the standard of a production-level system because the data-field safeguards found in production-level systems were not installed in the Yorktown intentionally," Rushton said.

"Those were things we accepted and we did what I consider to be a reasonable risk analysis," Rushton said. "If it appeared to compromise the safety of the crew, we didn't do it."

Slide 29: Озоновая дыра

Then, in 1974, two scientists wrote of a new concern that CFCs could potentially reduce levels of ozone in the stratosphere, the layer of atmosphere from 10 to 50 km in altitude. In 1975 the U.S. Congress asked NASA to develop a “comprehensive program of research, technology, and monitoring of phenomena of the upper atmosphere.” In particular, Congress’ intent was to ascertain the “health” of the ozone layer.

In 1978 Nimbus-7 carried two other new NASA sensors designed to measure the total amount of ozone in a given column of atmosphere over the entire globe—called the Solar Backscatter Ultraviolet (SBUV) instrument and the Total Ozone Mapping Spectrometer (TOMS). Sensitive to radiant energy in the ultraviolet region of the spectrum, these sensors took advantage of the fact that molecules and aerosol particles reflect certain wavelengths of ultraviolet rays while ozone absorbs others at different levels in the atmosphere. By analyzing the amount of ultraviolet energy reflected back up to the spacecraft, researchers could produce profiles of how thick or thin the ozone was at different altitudes and locations.

Ironically, it wasn't until October 1985 that a British team of scientists found a significant reduction in ozone over Halley Bay, Antarctica. Using a ground-based Dobson ozone spectrophotometer, the team found that the amount of stratospheric ozone over Halley Bay was about 40 percent less than it had been the previous year. Their finding stunned the science community because they were expecting anthropogenic ozone depletion to occur first at upper levels in the stratosphere (30 to 50 km) and so they anticipated that the initial signal of depletion in a total column of ozone would be weak.

NASA researchers hastily reviewed their TOMS data and found that it too had detected a dramatic loss of ozone over all of Antarctica. Why hadn't they discovered the phenomenon earlier? Unfortunately, the TOMS data analysis software had been programmed to flag and set aside data points that deviated greatly from expected measurements and so the initial measurements that should have set off alarms were simply overlooked.

In short, the TOMS team failed to detect the ozone depletion years earlier because it was much more severe than scientists expected.

Slide 30: Ракетный комплекс Patriot

During the first Persian Gulf war, Iraqi-fired Scud missiles were the most threatening airborne enemies to U.S. troops. Once one of these speeding death rockets launched, the U.S.'s best defense was to intercept it with an antiballistic Patriot missile. The Patriot worked a bit like a shotgun, getting within range of an oncoming missile before blasting out a cloud of 1,000 pellets to detonate its warhead.

A Patriot needed to deploy its pellets between 5 and 10 meters from an oncoming missile for the best results. This requires split-second timing, which is always tricky with two objects moving very fast toward each other. Even the Patriot's most prominent booster, then-President George H.W. Bush, conceded that one Scud (out of 42 fired) got past the Patriot. The single failure the president acknowledged was at a U.S. base in Dhahran, Saudi Arabia, on Feb. 25, 1991, and it cost 28 soldiers their lives. The fault was traced to a software error.

The Patriot's trajectory calculations revolved around the timing of radar pulses, and they had to be modified to deal with the high speed of modern missiles. A subroutine was introduced to convert clock time more accurately into floating-point figures for calculation. It was a neat kludge, but the programmers did not put the call to the subroutine everywhere it was needed. High-speed trajectories based on one accurately timed radar pulse and one less-precise time increased the chances of poorly timed deployment.

Apparently, the issue was known, and a temporary fix was in place: Reboot the system every so often to reset the clocks. Unfortunately, the term "every so often" wasn't defined, and that was the problem in late February at Dhahran. The system had been running for 100 hours, and the clocks were off by about a third of a second. A Scud travels half a kilometer in that time, so there was no chance the Patriot could have intercepted it.

On a side note, some experts did dispute the president's claims of a more than 97% success rate for Patriots vs. Scuds, so it's possible that this bug caused more (but less high-profile) damage than the incident at Dhahran.

Slide 31 & 32: Коллапс Харфорского Колизея

Cost: $70 million, plus another $20 million damage to the local economy

Disaster: Just hours after thousands of fans had left the Hartford Coliseum, the steel-latticed roof collapsed under the weight of wet snow.

Cause: The programmer of the CAD software used to design the coliseum incorrectly assumed the steel roof supports would only face pure compression. But when one of the supports unexpectedly buckled from the snow, it set off a chain reaction that brought down the other roof sections like dominoes. (more)

Collapse

On January 18, 1978 the Hartford Arena experienced the largest snowstorm of its five-year life. At 4:15 A.M. with a loud crack the center of the arena's roof plummeted the 83-feet to the floor of the arena throwing the corners into the air. Just hours earlier the arena had been packed for a hockey game. Luckily it was empty by the time of the collapse, and no one was hurt (Ross, 1984).

Causes of Failure

Hartford appointed a three-member panel to manage the investigation of the collapse. This panel in turn hired Lev Zetlin Associates, Inc. (LZA) to ascertain the cause of the collapse, and to propose a demolition procedure (Ross, 1984). LZA discovered that the roof began failing as soon as it was completed due to design deficiencies. A photograph taken during construction showed obvious bowing in two of the members in the top layer. Three major design errors coupled with the underestimation of the dead load by 20% (estimated frame weight = 18 psf, actual frame weight = 23 psf) allowed the weight of the accumulated snow to collapse the roof (*ENR*, April 6, 1978). The load on the day of collapse was 66-73 psf, while the arena should have had a design capacity of at least 140 psf (*ENR*, June 22, 1978). The three design errors responsible for the collapse are listed below.

- The top layer's exterior compression members on the east and the west faces were overloaded by 852%.
- The top layer's exterior compression members on the north and the south faces were overloaded by 213%.
- The top layer's interior compression members in the east-west direction were overloaded by 72%.

In addition to these errors in the original design, LZA discovered that the details omitted the midpoint braces for the rods in the top layer. The exterior rods were only braced every 30-feet, rather than the 15-feet intervals specified, and the interior rods were only partially and insufficiently braced at their midpoints. This significantly reduced the load that the roof could safely carry. The table below compares some of original details to actual designs used in the building, demonstrating the reduction in strength that these changes caused. Connection A was typically used on the east-west edges of the roof, while connection B was used on the north-south edges. Most of the interior bars used connection C, while a few used connection D. The

key difference between the original and the as-built details is that the diagonal members were attached some distance below the horizontal members, and thus were unable to brace the horizontal members against buckling.

Slide 33: Баг ‘конец света’ (почти)

Remember how the world descended into nuclear oblivion on Sept. 23, 1983? No? Well, thank your lucky stars -- this is a tale of bugs so major they could have brought the entire world to a standstill.

It was all averted by the common sense of one individual, who ignored the Soviet early-warning system's faulty reports of incoming missiles and didn't launch a counterattack on the United States.

The warning system set off klaxons at half past midnight on that September morning. Apparently, the U.S. had launched five nuclear missiles toward what the U.S. president had taken to calling "the Evil Empire."

At the time, Lt. Col. Stanislaus Petrov reasoned his way to a decision not to respond: The USSR was in a shouting match with the U.S. about a Soviet attack on Korean Air Lines Flight 007 three weeks earlier, but it was only a rhetorical battle at that stage. Besides, if the U.S. wanted to attack the Soviet Union, would it really launch only five missiles?

Petrov ordered his men to stand down, and 15 minutes later, radar outposts confirmed that there were no incoming missiles. The decision took less than five minutes, it was confirmed within half an hour, and the world remained at peace.

When the early-warning system was later analyzed, it was found to have more bugs than a suburban compost heap -- which meant that although Stanislaus Petrov had saved the world, he'd made a serious error of judgment: He had shown up the incompetence of Soviet programmers.

This was not good for morale, or for the lieutenant colonel. He was cold-shouldered into an early retirement and was largely unsung until May 21, 2004, when a San Francisco-based organization called the Association of World Citizens bestowed its highest honor -- world citizenship -- and a financial reward on him.

Slide 34: Авария в энергосистеме в США и Канаде (2003)

14 августа 2003 года между 15:45 и 16:15 по стандартному восточному времени, наблюдатели в Кливленде, Толедо, городе Нью-Йорк, Олбани, Детройте и в части Нью-Джерси сообщили о перебоях в подаче электроэнергии [1]. Позже последовали проблемы в изначально не затронутых регионах, включая все 5 районов города Нью-Йорк и в части Лонг-Айленда, округе Вестчестер, штатах Нью-Джерси, Вермонта и Коннектикута и большей части юга провинции Онтарио, включая Торонто [2].

Около 10 млн человек в Канаде (примерно треть населения) и 40 млн — в США остались без электричества. Однако большинство жизненно важных служб продолжали работать.

Закрылись многие аэропорты, включая международный аэропорт Пирсона в Торонто и все аэропорты Нью-Йорка. Во многих местах, включая Торонто и Нью-Йорк, прекратило работу метро. Застрявших в метро пассажиров пришлось эвакуировать. В отдельных местах, в частности в Детройте, были перебои с водой. Мобильные телефоны работали очень плохо, с большими перебоями, но стационарная телефонная связь продолжала функционировать. В результате у телефонных автоматов на улицах выстроились огромные очереди. Провайдеры продолжали работать, но в условиях отсутствия электричества единственным способом входа в интернет во многих местах осталось соединение через ноутбуки, работающие от аккумуляторов. Поскольку транзисторных приёмников у многих жителей США и Канады не было, они остались без возможности узнать новости. Жара в Нью-Йорке достигла 33 градуса в тени, но кондиционеры и вентиляторы не работали.

Денежный ущерб составил 6 миллиардов долларов.

Причины аварии, как выяснилось, следующие [3]:

- Деревья под линиями электропередачи периодически подстригают. В данном случае это не было сделано.
- Из-за очень высокого потребления электроэнергии, линии электропередачи в Кливленде, Огайо, нагрелись, провисли (из-за теплового расширения проводов) и коснулись деревьев. Произошло короткое замыкание.
- Электростанция в Кливленде вышла из строя.
- Из-за ошибки в компьютерной системе, а также из-за нехватки персонала, другие центры управления не были извещены.
- Произошло цепное отключение около 100 других электростанций.

Statements made in the aftermath

During the first two hours of the event, various officials offered speculative explanations as to its root cause:

- Official reports from the office of Canadian Prime Minister Jean Chrétien stated that lightning had struck a power plant in northern New York, resulting in a cascading failure of the surrounding power grid and wide-area electric power transmission grid. However, power officials in the State of New York responded by stating that the problem did not

originate in the United States, that there was no rain storm in the area where the lightning strike was supposed to have taken place, and that the power plant in question remained in operation throughout the blackout.

- Canadian Defence Minister John McCallum blamed an outage at a nuclear plant in Pennsylvania, but that state's authorities reported that all the plants were functioning normally. McCallum later stated that his sources had given him incorrect information.
- New York state Governor George Pataki blamed the power outage on Canada, stating "the New York independent systems operator says they are virtually certain it had nothing to do in New York state. And they believe it occurred west of Ontario, cascaded from there into Ontario, Canada, and through the northeast."[9] This was later proven to be false.
- CNN cited unnamed officials as saying that the Niagara-Mohawk power grid, which provides power for large portions of New York and parts of Canada, was overloaded. Between 4:10 and 4:13 p.m. EDT, 21 power stations throughout that grid shut down.[6]
- New Mexico governor Bill Richardson, who formerly headed the Department of Energy, in a live television interview 2 hours into the blackout characterized the United States as "a superpower with a third-world electricity grid." In Europe, this statement was published accompanied with comparisons highlighting the tighter, safer and better interconnected European electricity network (though it would suffer a similar blackout six weeks later).
- In the ensuing days, critics focused on the role of electricity market deregulation for the inadequate state of the electric power transmission grid, claiming that deregulation laws and electricity market mechanisms have failed to provide market participants with sufficient incentives to construct new transmission lines and maintain system security. [*who?*]
- Later that night, claims surfaced that the blackout may have started in Ohio up to one hour before the network shut down, a claim denied by Ohio's FirstEnergy utility.[*citation needed*]
- The president of the North American Electric Reliability Corporation said that the problem originated in Ohio.[10]
- By Saturday morning, investigators believed that the problem began with a sudden shift in the direction of power flow on the northern portion of the Lake Erie Transmission Loop, a system of transmission lines that circles Lake Erie on both U.S. and Canadian soil.[*citation needed*]

Causes

Background

Electrical power cannot easily be stored over extended periods of time, and is generally consumed less than a second after being produced. The load on any network must be matched by the supply to it and its ability to transmit that power. Any overload of a power line, or underload/overload of a generator, can cause hard-to-repair and costly damage, so the affected device is disconnected from the network if a serious imbalance is detected.

As power lines carry more current, they get hotter. This causes them to lengthen and sag between towers. They may safely reach a specified minimum clearance height above the ground. If the lines sag further, a flashover to nearby objects (such as trees) can occur, causing a transient increase in current. Automatic protective relays detect the high current and quickly act to disconnect the faulted line from service. To maintain the lines' specified operating clearance, the right-of-way must be kept clear of vegetation.

Should a fault occur and take a line out of service, the change in current flow is compensated by other transmission lines, which must have enough spare capacity to carry the excess current. If they do not, overload protection in those lines will also trip, causing a cascading failure as the excess current is switched onto neighbouring circuits running at or near their capacity.

System operators are responsible for ensuring that power supply and loads remain balanced, and for keeping the system within safe operational limits such that no single fault can cause the system to fail. After a failure affecting their system, operators must obtain more power from generators or other regions or "shed load" (meaning to intentionally cut power to some areas, or reducing voltage to a given area, creating a brownout) until they can be sure that the worst remaining possible failure anywhere in the system will not cause a system collapse. In an emergency, they are expected to immediately shed load as required to bring the system into balance.

To assist the operators there are computer systems, with backups, which issue alarms when there are faults in the transmission or generation system. Power flow modeling tools let them analyze the current state of their network, predict whether any parts of it may be overloaded, and predict what the worst possible failure left is, so that they can change the distribution of generation or reconfigure the transmission system to prevent a failure should this situation occur. If the computer systems and their backups fail, the operators are required to monitor the grid manually, instead of relying on computer alerts. If they cannot interpret the current state of the power grid in such an event, they follow a contingency plan, contacting other plant and grid operators by telephone if necessary. If there is a failure, they are also required to notify adjacent areas which may be affected, so those can predict the possible effects on their own systems.

Local operators are co-ordinated by regional centers, but the operating principle is the same whether the network is large or small.

Findings

In February 2004, the U.S.-Canada Power System Outage Task Force released their final report, placing the causes of the blackout into four groups:[12]

First, that FirstEnergy and its reliability council "failed to assess and understand the inadequacies of FE’s system, particularly with respect to voltage instability and the vulnerability of the Cleveland-Akron area, and FE did not operate its system with appropriate voltage criteria".

Second, that FirstEnergy "did not recognize or understand the deteriorating condition of its system".

Third, that FirstEnergy "failed to manage adequately tree growth in its transmission rights-of-way".

Finally, the "failure of the interconnected grid’s reliability organizations to provide effective real-time diagnostic support."

The report states that a generating plant in Eastlake, Ohio (a suburb of Cleveland) went offline amid high electrical demand, putting a strain on high-voltage power lines (located in a distant rural setting) which later went out of service when they came in contact with "overgrown trees". The cascading effect that resulted ultimately forced the shutdown of more than 100 power plants.

A preliminary report says four or five capacitor banks in the Cleveland-Akron area were removed from service for government inspection, including capacitor banks at Fox and Avon 138-kV substations. These reactive power sources are important for voltage support, but were not restored to service that afternoon despite the system operators' need for more reactive power compensation. The normal practice is to inspect in off-peak seasons, but government officials demanded the inspection when inspectors were available. The final report does not mention this government demand. The lack of reactive power compensation caused the protective relay trip that brought the system down.[*citation needed*] The same thing also caused the 1965 blackout.

This trip caused overloading of other transmission lines, tripping their relays. Once these multiple trips occurred, multiple generators suddenly lost parts of their loads, so they accelerated out of phase with the grid at different rates, and tripped out to prevent damage.

Computer failure

A software bug known as a race condition existed in General Electric Energy's Unix-based XA/21 energy management system. Once triggered, the bug stalled FirstEnergy's control room alarm system for over an hour. System operators were unaware of the malfunction; the failure deprived them of both audio and visual alerts for important changes in system state.[13][14] After the alarm system failure, unprocessed events queued up and the primary server failed within 30 minutes. Then all applications (including the stalled alarm system) were automatically transferred to the backup server, which itself failed at 14:54. The server failures slowed the screen refresh rate of the operators' computer consoles from 1–3 seconds to 59 seconds per screen. The lack of alarms led operators to dismiss a call from American Electric Power about the tripping and reclosure of a 345 kV shared line in northeast Ohio. Technical support informed control room personnel of the alarm system failure at 15:42.[15]

Sequence of events

The following is the blackout's sequence of events on August 14, 2003[16][17][18] (times in EDT):

- 12:15 p.m. Incorrect telemetry data renders inoperative the state estimator, a power flow monitoring tool operated by the Indiana-based Midwest Independent Transmission System Operator (MISO). An operator corrects the telemetry problem but forgets to restart the monitoring tool.
- 1:31 p.m. The Eastlake, Ohio generating plant shuts down. The plant is owned by FirstEnergy, an Akron, Ohio-based company that had experienced extensive recent maintenance problems.[*specify*]
- 2:02 p.m. The first of several 345 kV overhead transmission lines in northeast Ohio fails due to contact with a tree in Walton Hills, Ohio.[19] 41°21′22″N 81°34′10″W[*citation needed*]
- 2:14 p.m. An alarm system fails at FirstEnergy's control room and is not repaired.
- 3:05 p.m. A 345 kV transmission line known as the Chamberlin-Harding line fails in Parma, south of Cleveland, due to a tree.
- 3:17 p.m. Voltage dips temporarily on the Ohio portion of the grid. Controllers take no action.
- 3:32 p.m. Power shifted by the first failure onto another 345 kV power line, the Hanna-Juniper interconnection, causes it to sag into a tree, bringing it offline as well. While MISO and FirstEnergy controllers concentrate on understanding the failures, they fail to inform system controllers in nearby states.
- 3:39 p.m. A FirstEnergy 138 kV line fails in northern Ohio.[20]

- 3:41 p.m. A circuit breaker connecting FirstEnergy's grid with that of American Electric Power is tripped as a 345 kV power line (Star-South Canton interconnection) and fifteen 138 kV lines fail in rapid succession in northern Ohio.
- 3:46 p.m. A fifth 345 kV line, the Tidd-Canton Central line, trips offline.
- 4:05:57 p.m. The Sammis-Star 345 kV line trips due to undervoltage and overcurrent interpreted as a short circuit. Later analysis suggests that the blackout could have been averted prior to this failure by cutting 1.5 GW of load in the Cleveland–Akron area.
- 4:06–4:08 p.m. A sustained power surge north toward Cleveland overloads three 138 kV lines.
- 4:09:02 p.m. Voltage sags deeply as Ohio draws 2 GW of power from Michigan, creating simultaneous undervoltage and overcurrent conditions as power attempts to flow in such a way as to rebalance the system's voltage.
- 4:10:34 p.m. Many transmission lines trip out, first in Michigan and then in Ohio, blocking the eastward flow of power around the south shore of Lake Erie from Toledo, Ohio, east through Erie, Pennsylvania and into the Buffalo, New York metropolitan area. Suddenly bereft of demand, generating stations go offline, creating a huge power deficit. In seconds, power surges in from the east, overloading east-coast power plants whose generators go offline as a protective measure, and the blackout is on.
- 4:10:37 p.m. The eastern and western Michigan power grids disconnect from each other. Two 345 kV lines in Michigan trip. A line that runs from Grand Ledge to Ann Arbor known as the Oneida-Majestic interconnection trips. A short time later, a line running from Bay City south to Flint in Consumers Energy's system known as the Hampton-Thetford line also trips.
- 4:10:38 p.m. Cleveland separates from the Pennsylvania grid.
- 4:10:39 p.m. 3.7 GW power flows from the east along the north shore of Lake Erie, through Ontario to southern Michigan and northern Ohio, a flow more than ten times greater than the condition 30 seconds earlier, causing a voltage drop across the system.
- 4:10:40 p.m. Flow flips to 2 GW eastward from Michigan through Ontario (a net reversal of 5.7 GW of power), then reverses back westward again within a half second.
- 4:10:43 p.m. International connections between the United States and Canada begin failing.
- 4:10:45 p.m. Northwestern Ontario separates from the east when the Wawa-Marathon 230 kV line north of Lake Superior disconnects. The first Ontario power plants go offline in response to the unstable voltage and current demand on the system.
- 4:10:46 p.m. New York separates from the New England grid.
- 4:10:50 p.m. Ontario separates from the western New York grid.
- 4:11:57 p.m. The Keith-Waterman, Bunce Creek-Scott 230 kV lines and the St. Clair-Lambton #1 230 kV line and #2 345 kV line between Michigan and Ontario fail.
- 4:12:03 p.m. Windsor, Ontario and surrounding areas drop off the grid.
- 4:12:58 p.m. Northern New Jersey separates its power-grids from New York and the Philadelphia area, causing a cascade of failing secondary generator plants along the New Jersey coast and throughout the inland regions west.
- 4:13 p.m. End of cascading failure. 256 power plants are off-line, 85% of which went offline after the grid separations occurred, most due to the action of automatic protective controls.

Slide 35: ‘Червяк’ Морриса

November 3, 1988 is already coming to be known as Black Thursday. System administrators around the country came to work on that day and discovered that their networks of computers were laboring under a huge load. If they were able to log in and generate a system status listing, they saw what appeared to be dozens or hundreds of ‘‘shell’’ (command interpreter) processes. If they tried to kill the processes, they found that new processes appeared faster than they could kill them. Rebooting the computer seemed to have no effect—within minutes after starting up again, the machine was overloaded by these mysterious processes.

These systems had been invaded by a worm. A worm is a program that propagates itself across a network, using resources on one machine to attack other machines. (A worm is not quite the same as a virus, which is a program fragment that inserts itself into other programs.) The worm had taken advantage of lapses in security on systems that were running 4.2 or 4.3 BSD UNIX or derivatives like SunOS. These lapses allowed it to connect to machines across a network, bypass their login authentication, copy itself and then proceed to attack still more machines. The massive system load was generated by multitudes of worms trying to propagate the epidemic.

11/2: 1800 (approx.)
This date and time were seen on worm files found on *prep.ai.mit.edu*, a VAX 11/750 at the MIT Artificial Intelligence Laboratory. The files were removed later, and the precise time was lost. System logging on *prep* had been broken for two weeks. The system doesn’t run accounting and the disks aren’t backed up to tape: a perfect target. A number of ‘‘tourist’’ users (individuals using public accounts) were reported to be active that evening. These users would have appeared in the session logging, but see below.
11/2: 1824 First known West Coast infection: *rand.org* at Rand Corp. in Santa Monica.
11/2: 1904 *csgw.berkeley.edu* is infected. This machine is a major network gateway at UC Berkeley. Mike Karels and Phil Lapsley discover the infection shortly afterward.
11/2: 1954 *mimsy.umd.edu* is attacked through its *finger* server. This machine is at the University of Maryland College Park Computer Science Department.
11/2: 2000 (approx.)
Suns at the MIT AI Lab are attacked.
11/2: 2028 First *sendmail* attack on mimsy.
11/2: 2040 Berkeley staff figure out the *sendmail* and *rsh* attacks, notice *telnet* and *finger* peculiarities, and start shutting these services off.
11/2: 2049 *cs.utah.edu* is infected. This VAX 8600 is the central Computer Science Department machine at the University of Utah. The next several entries follow documented events at Utah and are representative of other infections around the country.
11/2: 2109 First *sendmail* attack at *cs.utah.edu*.
11/2: 2121 The load average on *cs.utah.edu* reaches 5. The ‘‘load average’’ is a system-generated value that represents the average number of jobs in the run queue over the last minute; a load of 5 on a VAX 8600 noticeably degrades response times, while a load over 20 is a drastic degradation. At 9 PM, the load is typically between 0.5 and 2.
11/2: 2141 The load average on *cs.utah.edu* reaches 7.
11/2: 2201 The load average on *cs.utah.edu* reaches 16.
11/2: 2206 The maximum number of distinct runnable processes (100) is reached on *cs.utah.edu*; the system is unusable.
11/2: 2220 Jeff Forys at Utah kills off worms on *cs.utah.edu*. Utah Sun clusters are infected.
11/2: 2241 Re-infestation causes the load average to reach 27 on *cs.utah.edu*.
11/2: 2249 Forys shuts down *cs.utah.edu*.
11/3: 2321 Re-infestation causes the load average to reach 37 on *cs.utah.edu*, despite continuous efforts by Forys to kill worms.
11/2: 2328 Peter Yee at NASA Ames Research Center posts a warning to the TCP-IP mailing list: ‘‘We are currently under attack from an Internet VIRUS. It has hit UC Berkeley, UC San Diego, Lawrence Livermore, Stanford, and NASA Ames.’’ He suggests turning off *telnet*, *ftp*, *finger*, *rsh* and SMTP services. He does not mention *rexec*. Yee is actually at Berkeley working with Keith Bostic, Mike Karels and Phil Lapsley.
11/3: 0034 At another’s prompting, Andy Sudduth of Harvard anonymously posts a warning to the TCP-IP list: ‘‘There may be a virus loose on the internet.’’ This is the first message that (briefly) describes how the

finger attack works, describes how to defeat the SMTP attack by rebuilding *sendmail*, and explicitly mentions the *rexec* attack. Unfortunately Sudduth's message is blocked at *relay.cs.net* while that gateway is shut down to combat the worm, and it does not get delivered for almost two days. Sudduth acknowledges authorship of the message in a subsequent message to TCP-IP on Nov. 5.
11/3: 0254 Keith Bostic sends a fix for *sendmail* to the newsgroup comp.bugs.4bsd.ucb-fixes and to the TCP-IP mailing list. These fixes (and later ones) are also mailed directly to important system administrators around the country.
11/3: early morning
The *wtmp* session log is mysteriously removed on *prep.ai.mit.edu*.
11/3: 0507 Edward Wang at Berkeley figures out and reports the *finger* attack, but his message doesn't come to Mike Karels' attention for 12 hours.
11/3: 0900 The annual Berkeley Unix Workshop commences at UC Berkeley. 40 or so important system administrators and hackers are in town to attend, while disaster erupts at home. Several people who had planned to fly in on Thursday morning are trapped by the crisis. Keith Bostic spends much of the day on the phone at the Computer Systems Research Group offices answering calls from panicked system administrators from around the country.
11/3: 1500 (approx.)
The team at MIT Athena calls Berkeley with an example of how the *finger* server bug works.
11/3: 1626 Dave Pare arrives at Berkeley CSRG offices; disassembly and decompiling start shortly afterward using Pare's special tools.
11/3: 1800 (approx.)
The Berkeley group sends out for calzones. People arrive and leave; the offices are crowded, there's plenty of excitement. Parallel work is in progress at MIT Athena; the two groups swap code.
11/3: 1918 Keith Bostic posts a fix for the *finger* server.
11/4: 0600 Members of the Berkeley team, with the worm almost completely disassembled and largely decompiled, finally take off for a couple hours' sleep before returning to the workshop.
11/4: 1236 Theodore Ts'o of Project Athena at MIT publicly announces that MIT and Berkeley have completely disassembled the worm.
11/4: 1700 (approx.)
A short presentation on the worm is made at the end of the Berkeley UNIX Workshop.
11/8: National Computer Security Center meeting to discuss the worm. There are about 50 attendees.

11/11: 0038 Fully decompiled and commented worm source is installed at Berkeley.

Architecture of the worm

Disk containing the source code for the Morris Worm held at the Boston Museum of Science.

According to its creator, the Morris worm was not written to cause damage, but to gauge the size of the Internet. However, the worm was released from MIT to disguise the fact that the worm originally came from Cornell. Additionally, the Morris worm worked by exploiting known vulnerabilities in Unix sendmail, finger, and rsh/rexec, as well as weak passwords.[2] Due to reliance on rsh (normally disabled on untrusted networks) it should not succeed on a recent, properly configured system.

A supposedly unintended consequence of the code, however, caused it to be more damaging: a computer could be infected multiple times and each additional process would slow the machine down, eventually to the point of being unusable. This would have the same effect as a fork bomb and crash the computer. The main body of the worm could only infect DEC VAX machines running 4BSD, and Sun-3 systems. A portable C "grappling hook" component of the worm was used to pull over (download) the main body, and the grappling hook could run on other systems, loading them down and making them peripheral victims.

The mistake

The critical error that transformed the worm from a potentially harmless intellectual exercise into a virulent denial of service attack was in the spreading mechanism. The worm could have determined whether to invade a new computer by asking if there was already a copy running. But just doing this would have made it trivially easy to kill; everyone could just run a process that would answer "yes" when asked if there was already a copy, and the worm would stay away. The defense against this was inspired by Michael Rabin's mantra, "Randomization." To compensate for this possibility, Morris directed the worm to copy itself even if the response is "yes", 1 out of 7 times.[3] This level of replication proved excessive and the worm spread rapidly, infecting some computers multiple times. Morris remarked, when he heard of the mistake, that he "should have tried it on a simulator first."

Effects of the worm

It is usually reported that around 6,000 major UNIX machines were infected by the Morris worm. Paul Graham has claimed[4] that

"I was there when this statistic was cooked up, and this was the recipe: someone guessed that there were about 60,000 computers attached to the Internet, and that the worm might have infected ten percent of them."

The U.S. GAO put the cost of the damage at $100,000–10,000,000.[5]

The Morris worm prompted DARPA to fund the establishment of the CERT/CC at Carnegie Mellon University to give experts a central point for coordinating responses to network emergencies.[6] Gene Spafford also created the Phage mailing list to coordinate a response to the emergency.

Robert Morris was tried and convicted of violating United States Code: Title 18 (18 U.S.C. § 1030), the Computer Fraud and Abuse Act.[7] in *United States v Morris*. After appeals he was sentenced to three years probation, 400 hours of community service, and a fine of $10,000.[8]

The Morris worm has sometimes been referred to as the "Great Worm", because of the devastating effect it had on the Internet at that time, both in overall system downtime and in psychological impact on the perception of security and reliability of the Internet. The name was derived from the "Great Worms" of Tolkien: Scatha and Glaurung.[9]

Extra data

15. Y2K (1999)

Cost: $500 billion

Disaster: One man's disaster is another man's fortune, as demonstrated by the infamous Y2K bug. Businesses spent billions on programmers to fix a glitch in legacy software. While no significant computer failures occurred, preparation for the Y2K bug had a significant cost and time impact on all industries that use computer technology.

Cause: To save computer storage space, legacy software often stored the year for dates as two digit numbers, such as "99″ for 1999. The software also interpreted "00″ to mean 1900 rather than 2000, so when the year 2000 came along, bugs would result. (more)

---

17. Love Virus (2000)

Cost: $8.75 billion, millions of computers infected, significant data loss

Disaster: The LoveLetter worm infected millions of computers and caused more damage than any other computer virus in history. The worm deleted files, changed home pages and messed with the Registry.

Cause: LoveLetter infected users via e-mail, Internet chat and shared file systems. The email had an executable file attachment and subject line, "ILOVEYOU." When the user opened the attachment, the virus would infect the user's computer and send itself to everyone in the address book. (more)

No 'sorry' from Love Bug author

Five years on, Spyder keeps schtum

By Peter Hayes

Posted in Malware, 11th May 2005 13:09 GMT

Five years ago, a new "supervirus" hit the headlines. It had the two successful - but evil - elements: destructive virus coding coupled to an enticing title and the simple fact that it arrived from someone the recipient knew. The combination was virus dynamite.

Most viruses start slowly and then build power; "I Love You" hit the computer world like a bomb - anti-virus companies had not seen anything like it and while they struggled to contain the infection, copycats were re-titling the virus and releasing in their own language.

At this point you might expect the story to flash-pan to a prison cell and a description of the miscreant responsible for the outrage safely behind bars. However, the person who almost certainly wrote the virus - proved not only by his own admission but also by a stack of corroborating evidence - is today a free man with a no criminal record.

That's because in the Philippines - where he lives - there were no laws against computer misuse and the authorities had nothing to charge him with.

Today, almost five years after the event "Spyder" (real name Reomel Lamores) is saying nothing about the virus, referring all calls to his lawyer who - in turn - also refuses to comment. Not even "sorry" for the hundreds of millions of pounds of damage it allegedly caused and the general pandemonium it generated.

US tabloid TV programmes and book authors have dangled cheques in front of his nose - but at the moment he rejects them all. Local reports say he fears being kidnapped and has nightmares about being bundled on to a boat and taken to the USA.

Spyder's Web

In May 2000 Spyder was a minor computer programmer in the employ of the local China Bank, living in a low-rent Manila apartment with girlfriend Irene De Guzman. After its release into the wild, I Love You - aka "The Love Bug" - was quickly traced back to Spyder who was held by the authorities on unspecified grounds. US and European law enforcement authorities fought to be the first to try the then twenty seven year-old. The FBI even put seven men on the case, including their specialist virus sniffer Federick Bjorck.

Under questioning Spyder started by claiming total ignorance of events and blandly refused to assist the authorites. Even in the face of mounting evidence - including his own email address carrying the outbound virus - for which he had no explanation.

Eventually, he changed his story to the one he maintains to this day: The honest accident. He was messing around with coding "and the code escaped". Strangely this is slightly supported by the evidence. A thinking virus writer would have worked harder to cover his tracks. Some speculate that the whole stunt was created to impress his new girlfriend and he secretly hoped to get caught.

The virus was smart - for that time - in that it knew about file length. The full title (of the original e-mail) was LOVE-LETTER-FOR-YOU.TXT.vbs. The length of this tile was vital because (on default Windows setting) this hides the .vbs extension and it could be taken as plain text.

When up and running, the virus looked in the address book of Microsoft Outlook and sent copies of itself to everyone therein. For good measure, the virus then linked to four pages on Sky Internet (in the Philippines) which, in turn, downloaded the falsely named WIND-BUGFIX.exe. This had the effect of collecting and sending email addresses and passwords to a known second email address.

This second part of the operation didn’t last long. The ISP noted the huge surge in traffic and suspended the pages. Within hours the FBI bloodhounds were on the scent of the perpetrator. However before they did the title had already changed to one of the hundreds of variations that followed - Very Funny Joke.

In June of that year, and barring any other law with which to prosecute him, authorities charged Spyder's girlfriend Guzman - who came under suspicion because of a certain expertise with computers - under the local "Access Devices Act" of 1994, which outlaws the illegal use of account numbers and passwords - a law directly related to credit card fraud. The charges were

based on her owning the central computer from which the virus eminated. However, even these had to be dropped.

Later that year, the Philippines introduced new laws to target and outlaw a wide range of cybercrimes. But as the FBI are quick to point out - there are plenty of places left in the world that the cyber criminal is free to go about his or her business unhindered by the in-this-case-not-so-long arm of the law. ®

Honorable Mention #1 - LA Airport Flights Grounded (2007)

A single faulty piece of embedded software, on a network card, sends out faulty data on the United States Customs and Border Protection network, bringing the entire system to a halt. Nobody is able to leave or enter the U.S. from the LA Airport for over eight hours.
Result - Over 17,000 planes grounded for the duration of the outage

Honorable Mention #2 - The Ping of Death (1995)

A lack of error handling in the IP fragmentation reassembly code makes it possible to crash many Windows, Macintosh, and Unix operating systems by sending a malformed “ping” packet from anywhere on the Internet.
Result - The blue screen of death and giggling teenage hackers all over the nation.

Например, в 1992 году большой резонанс получил произошедший в Англии случай, когда "пошел в разнос" компьютер на станции скорой помощи: причина - неожиданно проявившиеся трудности с синхронизацией процессов в условиях большого количества поступивших заявок.

Согласно легенде, ранние конструкции торпед имели устройства безопасности, предохраняющие подводную лодку от повреждения, когда торпеда была запущена. Торпеда была сконструирована так, чтобы самоуничтожение запускалось, если торпеда поворачивалась на 180? . Идея была в том, что повернувшаяся на 180? торпеда может повредить выпустившую ее лодку. Однажды, капитан подводной лодки решил выпустить торпеду. Однако торпеда застряла в пусковой камере, и ее не смогли удалить. Капитан решил вернуться в порт для ремонта. Когда субмарина развернулась на 180 градусов, торпеда взорвалась и потопила лодку.

В данном случае ошибка заключалась в конструкции торпеды. Когда-то кто-то решил, что неплохо было бы встроить в торпеду некий предохранитель так, чтобы она не смогла потопить запустившую ее лодку. Идея была хорошей, а вот ее реализация - нет. Конструктор не учел тот самый случай, который и потопил субмарину. Однако остается неясным, на самом ли деле на заре конструирования вооружения для субмарин такой случай произошел или это всего лишь легенда. Но он показался нам достаточно реалистичным и поучительным, так что мы решили его привести.

В Афганистане двое наводчиков-наблюдателей (канадцы) подсвечивали цель для наведения на нее бомбы. После сброса бомбы в GPS приемнике закончились батарейки. Расчет их быстро заменил. В результате ракета прилетела не туда. Причина была в том, что после подачи питания в прибор переменные, отвечающие за координаты цели,

автоматически инициализировались координатами текущего местоположения. Наводчики погибли от близкого разрыва.

На испытаниях Су-24 регулярно случался отказ аппаратуры бомбометания. Причем происходило это только в том случае, если на цель заходил летчик-испытатель Ильюшин. Причина оказалось тоже не сложной. Только он заходил на цель с точностью, превышавшей машинную точность. Получался "машинный нуль", после чего шел сбой из-за попытки деления на ноль.
Когда испытывали ракету, прикрывавшую самолёт с хвоста, она аккуратно разворачивалась и настигала выпустивший её самолёт.

Начали выяснять. Причина оказалась элементарной: ракета снабжена стабилизаторами.

Стабилизаторы нужны, чтобы удерживать ракету передним концов в сторону вектора движения.

При вылете ракеты двигатель не сразу набирал мощность и какой-то момент она двигалась хвостом вперёд - естественно стабилизаторы разворачивали её в "правильном" направлении.

После этого врубался двигатель и ракета бодро летела к ближайшей цели.
Software problems in the automated baggage sorting system of a major airport in February 2008 prevented thousands of passengers from checking baggage for their flights. It was reported that the breakdown occurred during a software upgrade, despite pre-testing of the software. The system continued to have problems in subsequent months.
News reports in December of 2007 indicated that significant software problems were continuing to occur in a new ERP payroll system for a large urban school system. It was believed that more than one third of employees had received incorrect paychecks at various times since the new system went live the preceding January, resulting in overpayments of $53 million, as well as underpayments. An employees' union brought a lawsuit against the school system, the cost of the ERP system was expected to rise by 40%, and the non-payroll part of the ERP system was delayed. Inadequate testing reportedly contributed to the problems.

A software problem contributed to a rail car fire in a major underground metro system in April of 2007 according to newspaper accounts. The software reportedly failed to perform as expected in detecting and preventing excess power usage in equipment on a new passenger rail car, resulting in overheating and fire in the rail car, and evacuation and shutdown of part of the system.

A September 2006 news report indicated problems with software utilized in a state government's primary election, resulting in periodic unexpected rebooting of voter checkin machines, which were separate from the electronic voting machines, and resulted in confusion and delays at voting sites. The problem was reportedly due to insufficient testing.

In August of 2006 a U.S. government student loan service erroneously made public the personal data of as many as 21,000 borrowers on it's web site, due to a software error. The bug was fixed and the government department subsequently offered to arrange for free credit monitoring services for those affected.

In July 2004 newspapers reported that a new government welfare management system in Canada costing several hundred million dollars was unable to handle a simple benefits rate increase after being put into live operation. Reportedly the original contract allowed for only 6 weeks of acceptance testing and the system was never tested for its ability to handle a rate increase.

News reports in February of 2003 revealed that the U.S. Treasury Department mailed 50,000 Social Security checks without any beneficiary names. A spokesperson indicated that the missing names were due to an error in a software change. Replacement checks were subsequently mailed out with the problem corrected, and recipients were then able to cash their Social Security checks.

A newspaper columnist reported in July 2001 that a serious flaw was found in off-the-shelf software that had long been used in systems for tracking certain U.S. nuclear materials. The same software had been recently donated to another country to be used in tracking their own nuclear materials, and it was not until scientists in that country discovered the problem, and shared the information, that U.S. officials became aware of the problems.

In January of 2001 newspapers reported that a major European railroad was hit by the aftereffects of the Y2K bug. The company found that many of their newer trains would not run due to their inability to recognize the date '31/12/2000'; the trains were started by altering the control system's date settings.

A small town in Illinois in the U.S. received an unusually large monthly electric bill of $7 million in March of 1999. This was about 700 times larger than its normal bill. It turned out to be due to bugs in new software that had been purchased by the local power company to deal with Y2K software issues.

January 1998 news reports told of software problems at a major U.S. telecommunications company that resulted in no charges for long distance calls for a month for 400,000 customers. The problem went undetected until customers called up with questions about their bills.

A retail store chain filed suit in August of 1997 against a transaction processing system vendor (not a credit card company) due to the software's inability to handle credit cards with year 2000 expiration dates.

In November of 1996, newspapers reported that software bugs caused the 411 telephone information system of one of the U.S. RBOC's to fail for most of a day. Most of the 2000 operators had to search through phone books instead of using their 13,000,000-listing database. The bugs were introduced by new software modifications and the problem software had been installed on both the production and backup systems. A spokesman for the software vendor reportedly stated that 'It had nothing to do with the integrity of the software. It was human error.'

Software bugs caused the bank accounts of 823 customers of a major U.S. bank to be credited with $924,844,208.32 each in May of 1996, according to newspaper reports. The American Bankers Association claimed it was the largest such error in banking history. A bank spokesman said the programming errors were corrected and all funds were recovered.

---

xxx: Сделал в программе фичу, что при ошибке она отправляет письмо с подробностями и логом мне на почту. Отдал версию заказчику неделю назад.
xxx: На почту ничего до сих пор не пришло. Напрашивается вывод - заказчик программу не запускал.

Как-то 29 марта (это существенно) подходит ко мне молодой коллега Дима и говорит, что наша программа не пишет в БД. Я, ессно, спрашиваю "что менял" - отвечает "ничего". Ставим старую версию программы - не пишет. Ставим старую версию базы - не пишет. И

так продолжалось 3 дня. 1-го апреля подходит Дима и говорит, что программа стала писать в БД. Я думал первоапрельский розыгрыш. Пошёл проверил. Старая версия программы в старую версию БД пишет. Новая - тоже...
В итоге что выяснилось: эту программу писали давным-давно ещё под DOS. Дима же переделывал её под Windows. В DOS в структуре, в которой хранится дата, месяц считается от 0, а в windows'овской структуре - от единицы. При записи в базу старый формат даты преобразовывался в новый без учёта этой особенности. Таким образом получалось, что мы пытались записать в БД 29 февраля, затем 30-е, затем 31-е... СУБД отказывалась воспринимать эти даты, а вот первое марта - пожалуйста.
Как никто не заметил, что дата в БД отличается на месяц - не знаю

```
void initSomethingImportant()
{
   assert( init_very_important_lib() && "init library fail" );

   important_lib::start( "some_params" );
}
```

Еще вспомнилось, уже из древних времен. Если кто помнит, во времена доса и светлой памяти турбо С готового функционала для работы с мышью не было. Полагалось писать что-то вроде драйвера, вроде называлось это резидентной программой. Поскольку она могла получить управление в любой момент, ей полагалось перед работой сохранить значения регистров, а потом восстановить. Программка простая, пара десятков строк на ассемблере. Тоже из серии "написали и забыли". Ну и вот, после 2-3 лет жизни на 286-х компьютерах наступает светлая эпоха 386-го процессора. ОС и программы остаются 16-разрядными, основная программа на С++, поэтому замены одного ящика на другой я особо не замечаю. И вот в мышином резиденте остается сохранение\восстановление 16-разрядных регистров. При этом, программа в основном работала нормально (видимо, старшая часть регистров имела значение не часто). Но иногда ее начинало вдруг глючить. Зависело это, похоже, исключительно от расположения звезд: она могла работать целыми днями без эксцессов. Попытка выловить баг отладкой напоминала погоню за горизонтом: только вроде находишь место падения, ставишь туда точку прерывания, а она валится совсем в другом, где раньше работала. Жуть, в общем. Несколько дней я убила на погоню за тенью, пока случайно не вспомнила про мышиный драйвер и не посмотрела, как там регистры сохраняются.
