Lecture #23

Real Time Operating Systems

18-348 Embedded System Engineering

Philip Koopman

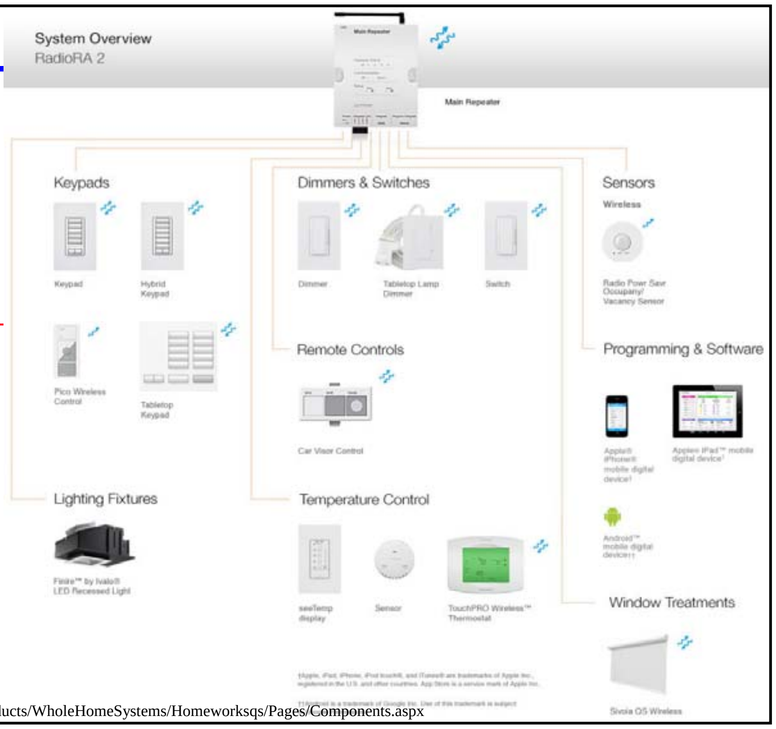
Monday, April 11, 2016

Electrical & Computer ENGINEERING

Carnegie Mellon

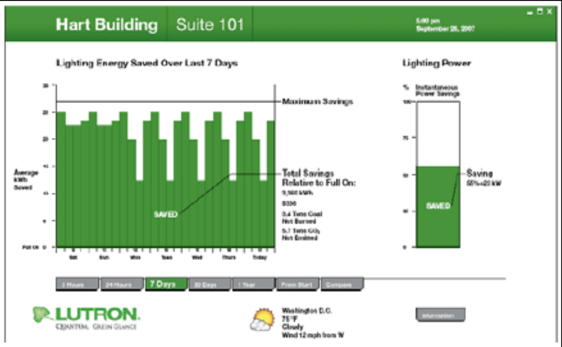
Lighting

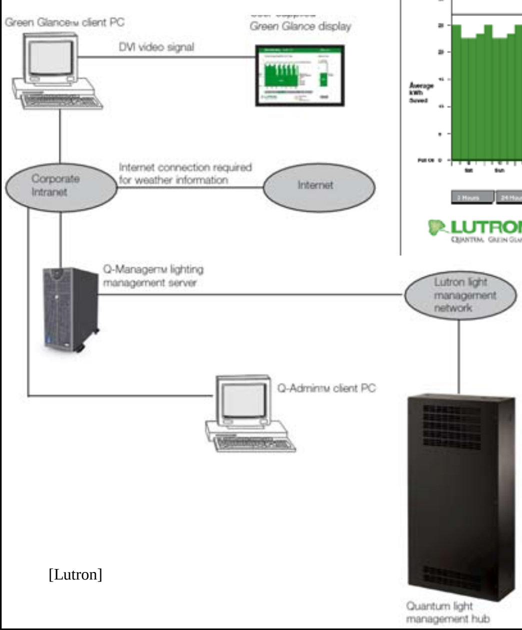
- ***It's just a light switch – how hard can it be?***



http://www.lutron.com/Products/WholeHomeSystems/Homeworksqs/Pages/Components.aspx

Energy Savings




Green Glance client PC
DVI video signal
Green Glance display
Corporate Intranet
Internet connection required for weather information
Internet
Q-Manager lighting management server
Lutron light management network
Q-Admin client PC
Quantum light management hub

[Lutron]

Real time light dimming

- Occupancy Sensors
- Daylight Sensors
- Wireless & wired networking
- Provide constant illumination across building
- Save by avoiding 100%-on

Where Are We Now?

- **Where we’ve been:**
  - Interrupts
  - Context switching and response time analysis
  - Concurrency
  - Scheduling
- **Where we’re going today:**
  - RTOS and other related topics
  - Priority inversion
  - Why software quality matters (safety & security)
- **Where we’re going next:**
  - Intro to embedded networks
  - System booting, control, safety
  - Test #2 on Wednesday April 20th, 2016

Preview

- **Priority Inversion**
  - Combining priorities with a mutex leads to complications
  - Priority inheritance & priority ceiling as solutions

- **RTOS overview**
  - What to look for in an RTOS
  - Market trends in RTOS
  - General embedded design trends

Remember the Major Scheduling Assumptions?

- **Five assumptions throughout this lecture**
  1. **Tasks {T_i} are perfectly periodic**
  2. **B=0**
  3. **$P_i = D_i$**
  4. **Worst case C_i**
  5. **Context switching is free**

Overcoming Assumptions

- **WHAT IF:**
  1. Tasks $\{T_i\}$ are NOT periodic
     – Use Sporadic techniques
  2. Tasks are NOT completely independent
     – Worry about dependencies
     (lets talk about this one)
  3. Deadline NOT = period
     – Use Deadline monotonic
  4. Worst case computation time c_i isn't known
     – Use worst case computation time, if known
     – Build or buy a tool to help determine Worst Case Execution Time (WCET)
     – Turn off caches and otherwise reduce variability in execution time
  5. Context switching is free (zero cost)
     – Gets messy depending on assumptions
     – Might have to include scheduler as task
     – Almost always need to account for blocking time B

Reminder: Basic Hazards

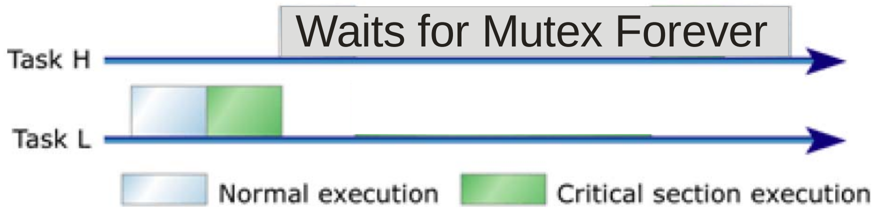
- **Deadlock**
  - Task A needs resources X and Y
  - Task B needs resources X and Y

  - Task A acquires mutex for resource X
  - Task B acquires mutex for resource Y

  - Task A waits forever to get mutex for resource Y
  - Task B waits forever to get mutex for resource X
- **Livelock**
  - Tasks release resources when they fail to acquire both X and Y, but… just keep deadlocking again and again
- **We're not to solve these here… desktop OS designers have these too**
  - But there are related priority problems specific to real time embedded systems

Mutex + Priorities Leads To Problems

- **Scenario: Higher priority task waits for release of shared resource**
  - Task L (low prio) acquires resource X via mutex
  - Task H (high prio) wants mutex for resource X and waits for it
- **Simplistic outcome with no remedies to problems (don't do this!)**
  - Task H hogs CPU in an infinite test-and-set loop waiting for resource X
  - Task L never gets CPU time, and never releases resource X
  - Strictly speaking, this is "starvation" rather than "deadlock"



[Renwick04] *modified*

Bounded Priority Inversion

- **An possible approach (BUT, this has problems…)**
  - Task H returns to scheduler every time mutex for resource X is busy
  - Somehow, scheduler knows to run Task L instead
    - If it is a round-robin preemptive scheduler, this will help
    - In prioritized scheduler, task H will have to reschedule itself for later
      - » Can get fancy with mutex release re-activating waiting tasks, whatever ….
  - Priority inversion is bounded – Task L will eventually release Mutex
    - And, if we keep critical regions short, this blocking time B won't be too bad

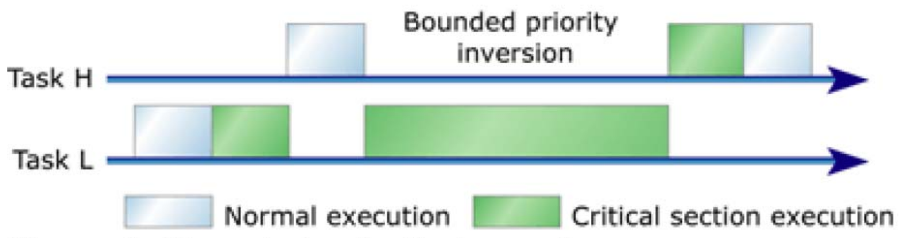


Figure 1: Bounded priority inversion

[Renwick04]

Unbounded Priority Inversion

- **But, simply having Task H relinquish the CPU isn't enough**
  - Task L acquires mutex X
  - Task H sees mutex X is busy, and goes to sleep for a while; Task L resumes
  - Task M preempts task L, and runs for a long time
  - Now task H is waiting for task M ➔ Priority Inversion
    - Task H is *effectively* running at the priority of task L because of this inversion

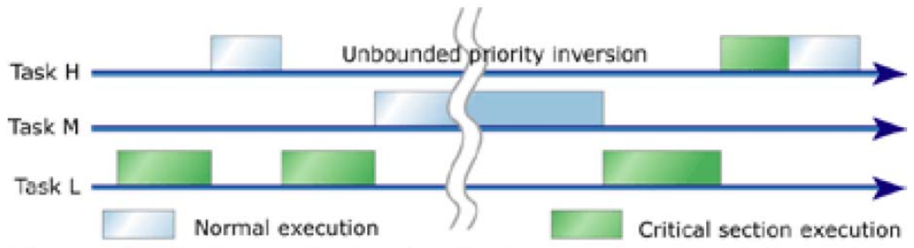


Figure 2: Unbounded priority inversion

[Renwick04]

Solution: Priority Inheritance

- **When task H finds a lock occupied:**
  - It elevates task L to at least as high a priority as task H
  - Task L runs until it releases the lock, but with priority of at least H
  - Task L is demoted back to its normal priority
  - Task H gets its lock as fast as possible; lock release by L ran at prio H
- **Idea: since mutex is delaying task H, free mutex as fast as you can**
  - Without suspending tasks having higher priority than H!
  - For previous slide picture, L would execute with higher prio than M

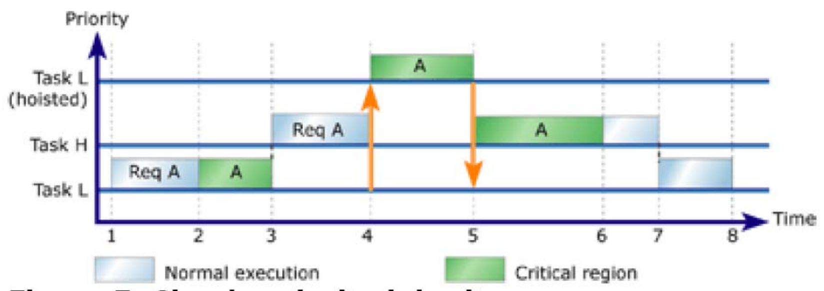


Figure 5: Simple priority inheritance

[Renwick04]

Priority Inheritance Pro/Con

- **Pro: it avoids many deadlocks and starvation scenarios!**
  - Only elevates priority when needed (only when high prio task wants mutex)
- **Run-time scheduling cost is perhaps neutral**
  - Task H burns up extra CPU time to run Task L at its priority
  - Blocking time B costs per the scheduling math are:
    - L runs at prio H, which effectively increases H's CPU usage
    - But, H would be "charged" with blocking time B regardless, so no big loss
- **Con: complexity can be high**
  - Almost-static priorities, not fully static
    - But, only changes when mutex encountered, not on every scheduling cycle
  - Nested priority elevations can be tricky to unwind as tasks complete
  - Multi-resource implementations are even trickier
- **If you can avoid need for a mutex, that helps a lot**
  - But sometimes you need a mutex; then you need priority inheritance too!

Mars Pathfinder Incident (Sojourner Rover)

- **July 4, 1997 – Pathfinder lands on Mars**
  - First US Mars landing since Vikings in 1976
  - First rover to land (vs. crash) on Mars
  - Uses VxWorks RTOS
- **But, a few days later…**
  - Multiple system resets occur
    - Watchdog timer saves the day!
    - System reset to safe state instead of unrecoverable crash
  - Reproduced on ground; patch uploaded to fix it
    - Developers didn't have Priority Inheritance turned on!
    - Scenario pretty much identical to H/M/L picture a couple slides back
    - Rough cause: "The data bus task executes very frequently and is time-critical -- we shouldn't spend the extra time in it to perform priority inheritance" [Jones07]

RTOS Selection

- **RTOS = Real Time Operating System**
  - An OS specifically intended to support real time scheduling
    - Usually, this means ability to meet deadlines
  - Can support any scheduling approach, but often is preemptive & prioritized
  - Usually designed to have low blocking time B

- **Why isn't plain Windows an RTOS?**
  - Example – Win NT (in all fairness, it was never supposed to be an RTOS!)
  - 31 priority levels (not enough if you need one per task and one per resource)
    - Round robin execution to all threads at same priority
    - Probably want 256 or more for an RTOS
  - Didn't support priority inheritance
  - Long blocking times on simple system calls (e.g., 670 usec+ on WinNT)
  - Device drivers aren't designed to guarantee minimum blocking time
  - Virtual memory is assumed active (swap to disk is a timing problem!)
  - It's expensive for mass market products at $186+ per license
  - Source: [http://www.dedicated-systems.com/magazine/97q2/winntasrtos.htm]

So What Do You Need In An RTOS?

Source: [Hawley03] Selecting a Real-Time Operating System, Embedded.com

- **Build vs. buy**
  - Don't build it if you can buy it ("free" = "buy" for right now)
  - More on this later

- **Footprint**
  - How much memory does the RTOS take?
  - Tasker can be very small, but there is more to an RTOS than that
  - Libraries
    - If you use one math function, does linker drag in all math functions?
    - Or can linker just link functions you actually use?
  - Feature subsetting
    - Can you get RTOS to include only features you need to minimize footprint?

RTOS Features – 2

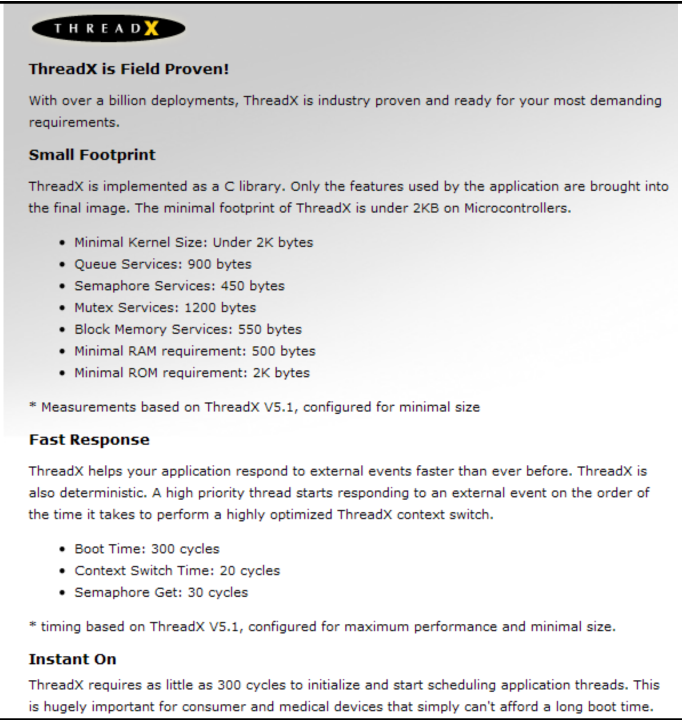
- **Performance**
  - Real Time != Real Fast … but Real Slow is no fun either
  - Blocking time B is key!
  - What is task switching time?
  - What is maximum blocking time within supplied code?
  - Does it get things such as device driver blocking right?
  - Boot time – does your customer want to wait 5 minutes to boot a flashlight?
  - Make sure you compare apples to apples – comparable CPUs and clock speeds
- **Add-ons**
  - Does it come with support for web connectivity?
  - Does it support domain-specific needs (e.g., MISRA C compiler for automotive?)
- **Tool support – comes with or supports other tools you need**
  - Compilers
  - Debuggers
  - Simulators, ICE, etc.

RTOS – 3

- **Standards support**
  - Windows?
  - POSIX (“Unix”)?
    - Watch out for subsetting! Might support some functions but not even a command prompt
    - QNX and RT-Linux have a command prompt
    - VxWorks is Posix compliant, but doesn’t support “fork”
  - Safety certification, if required (domain specific)
    - This is becoming more common for major players
- **Technical support**
  - Will they answer the phone at 3 AM if your biggest customer is down?
  - Training
  - Examples
- **Source code**
  - Some will provide you with source code outright so you can self-support
  - Some will put source code in escrow in case they go out of business

RTOS – 4

- **RTOS features you need**
  - Mutex / semaphore
    - Priority inheritance or priority ceiling
  - Scheduling support: RMS (big RTOS) or static multi-rate (medium RTOS) or single-rate cyclic exec (small RTOS)
  - Processes (big RTOS) or just tasks (medium/small RTOS)
  - Memory protection and memory management
- **Licensing – how much does it cost?**
  - Bulk license – flat fee for unlimited copies
  - Per-copy license – usually "runtime only" license is "cheap"
    - Development license may be expensive
  - Free software isn't really free
    - Support comes from somewhere – internal or 3rd party
- **Reputation**
  - Will the company be there for you?
    - Will it still be there tomorrow (is it one guy in a garage?)
  - Does its software actually work?



THREADX

ThreadX is Field Proven!

With over a billion deployments, ThreadX is industry proven and ready for your most demanding requirements.

Small Footprint

ThreadX is implemented as a C library. Only the features used by the application are brought into the final image. The minimal footprint of ThreadX is under 2KB on Microcontrollers.

- Minimal Kernel Size: Under 2K bytes
- Queue Services: 900 bytes
- Semaphore Services: 450 bytes
- Mutex Services: 1200 bytes
- Block Memory Services: 550 bytes
- Minimal RAM requirement: 500 bytes
- Minimal ROM requirement: 2K bytes

* Measurements based on ThreadX V5.1, configured for minimal size

Fast Response

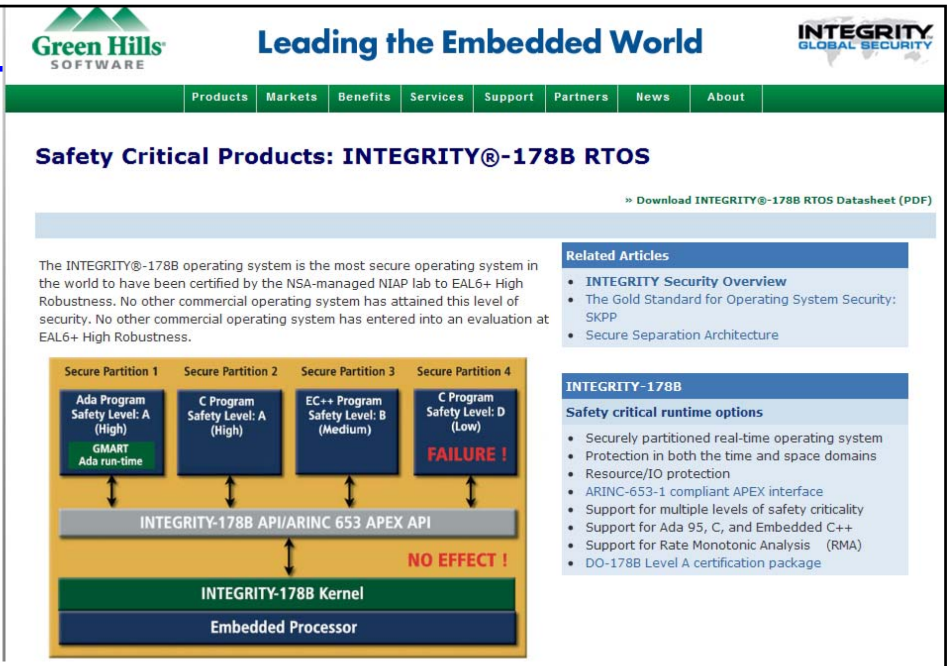
ThreadX helps your application respond to external events faster than ever before. ThreadX is also deterministic. A high priority thread starts responding to an external event on the order of the time it takes to perform a highly optimized ThreadX context switch.

- Boot Time: 300 cycles
- Context Switch Time: 20 cycles
- Semaphore Get: 30 cycles

* timing based on ThreadX V5.1, configured for maximum performance and minimal size.

Instant On

ThreadX requires as little as 300 cycles to initialize and start scheduling application threads. This is hugely important for consumer and medical devices that simply can't afford a long boot time.



Safety Critical Products: INTEGRITY®-178B RTOS

» Download INTEGRITY®-178B RTOS Datasheet (PDF)

The INTEGRITY®-178B operating system is the most secure operating system in the world to have been certified by the NSA-managed NIAP lab to EAL6+ High Robustness. No other commercial operating system has attained this level of security. No other commercial operating system has entered into an evaluation at EAL6+ High Robustness.

Secure Partition 1
Secure Partition 2
Secure Partition 3
Secure Partition 4
Ada Program Safety Level: A (High)
GMART Ada run-time
C Program Safety Level: A (High)
EC++ Program Safety Level: B (Medium)
C Program Safety Level: D (Low)
FAILURE !
INTEGRITY-178B API/ARINC 653 APEX API
NO EFFECT !
INTEGRITY-178B Kernel
Embedded Processor

Related Articles

- INTEGRITY Security Overview
- The Gold Standard for Operating System Security: SKPP
- Secure Separation Architecture

INTEGRITY-178B

Safety critical runtime options

- Securely partitioned real-time operating system
- Protection in both the time and space domains
- Resource/IO protection
- ARINC-653-1 compliant APEX interface
- Support for multiple levels of safety criticality
- Support for Ada 95, C, and Embedded C++
- Support for Rate Monotonic Analysis (RMA)
- DO-178B Level A certification package

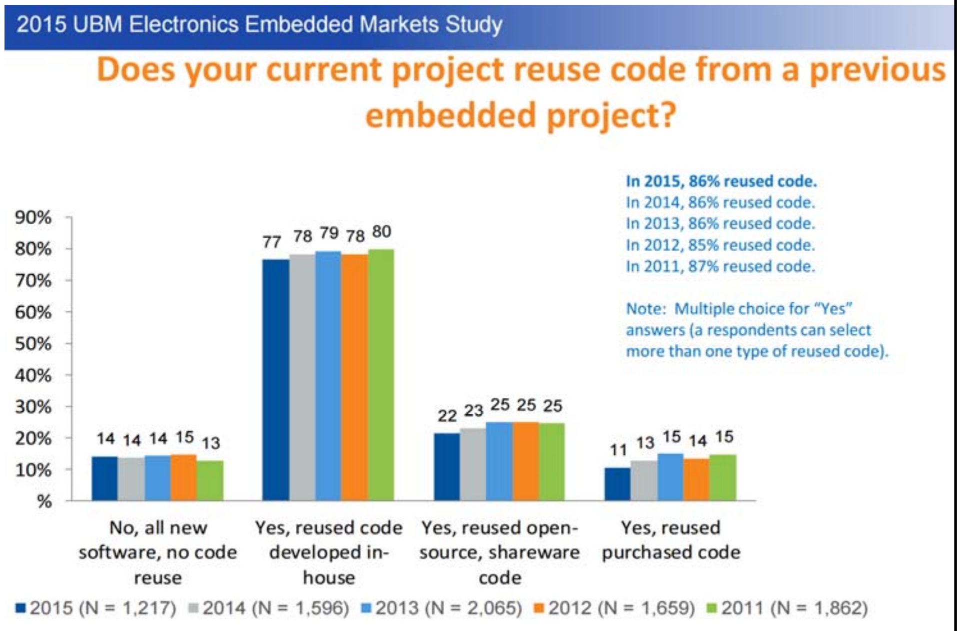
Adopting A Free RTOS Can Be Tricky

- **Example: Adopt a "free" RTOS**
  - Assume it's "free" (source code available), popular, and pretty good
  - Local engineers learn it and make some tweaks
  - Now you have your own local code base and some expert engineers
- **Is it really "free?"**
  - Engineers invested time learning it, but they'd do that for any RTOS
  - Local code base has to be maintained – this is *not* free
    - If bug fixes are published for initial code, have to adapt them to your version
    - Maybe no big deal if a small fraction of engineer's time
    - Engineer was good at RTOS design already, so it's a "free" skill
- **But what is the organizational cost?**
  - If that engineer leaves, you need to hire someone else with RTOS skills!
    - And convince them to move to whatever little town that company is in
  - May or may not be able to benefit from later add-on tools
    - May or may not be able to migrate to later major upgrades

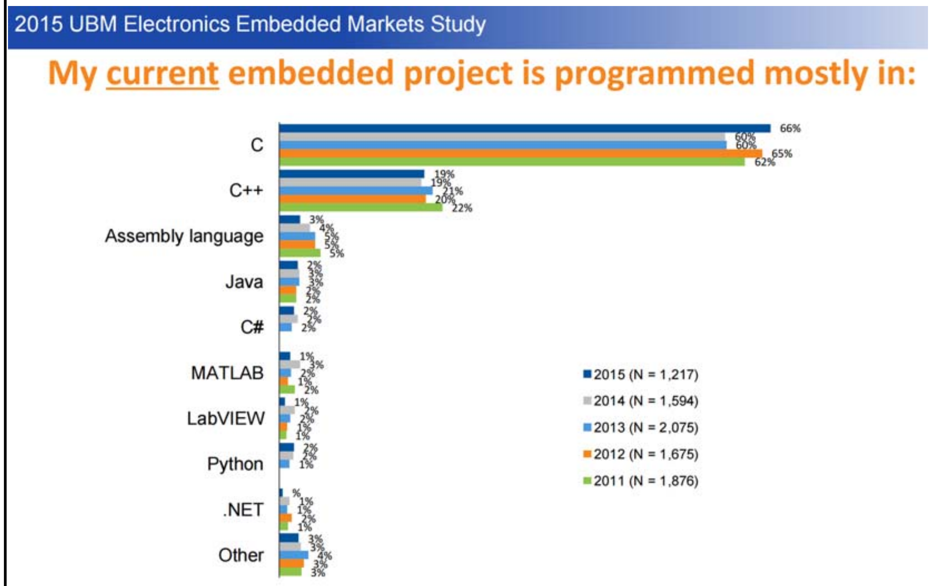
Industry Concern: Open Source "Poisoning"

- **Industry projects have to be *very* careful about open source**
  - Some open source licenses are no big deal (probably Berkeley)
  - Some open source licenses are *toxic* (especially GPL)
    - GPL library code and using compilers is OK; rest can be a problem
  - Some are in between
- **Common concerns with open source**
  - Requirement to publish source code of "derivative works"
  - Prohibition for fixed-function product "Tivo-ization" prohibited
  - Tracking and publishing copyright attribution (an annoyance)
  - Possibility of being sued for patent infringement by open source code
- **How do you manage the risks?**
  - Use open source tracking tools that sniff out all open source code in a build
  - Have explicit legal department sign-off on every open source component
    - Sometimes you can't use them because the legal issues are too tough
    - And sometimes it's OK … depends upon product & company

Few Projects Are "Clean Sheet of Paper"

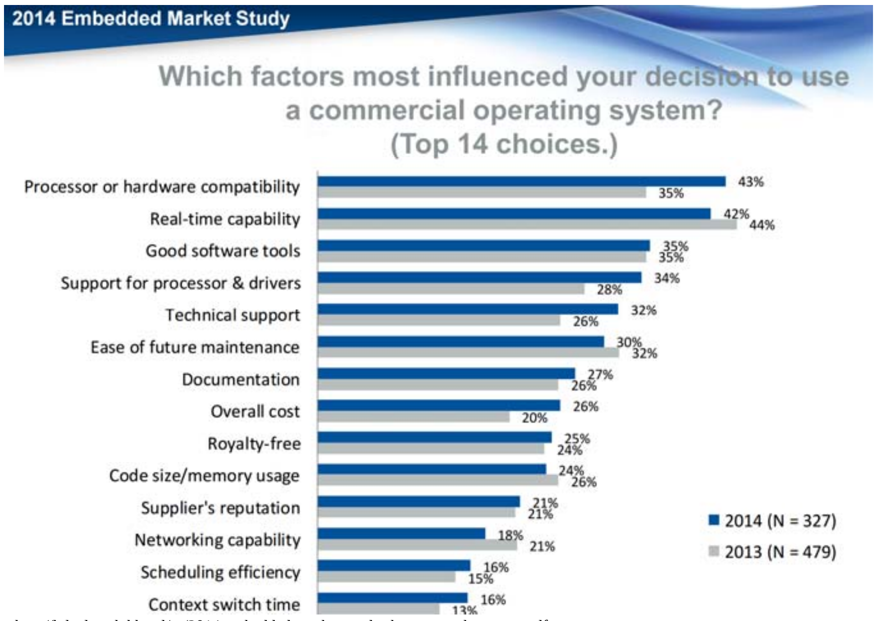


C & C++ Are Prevalent



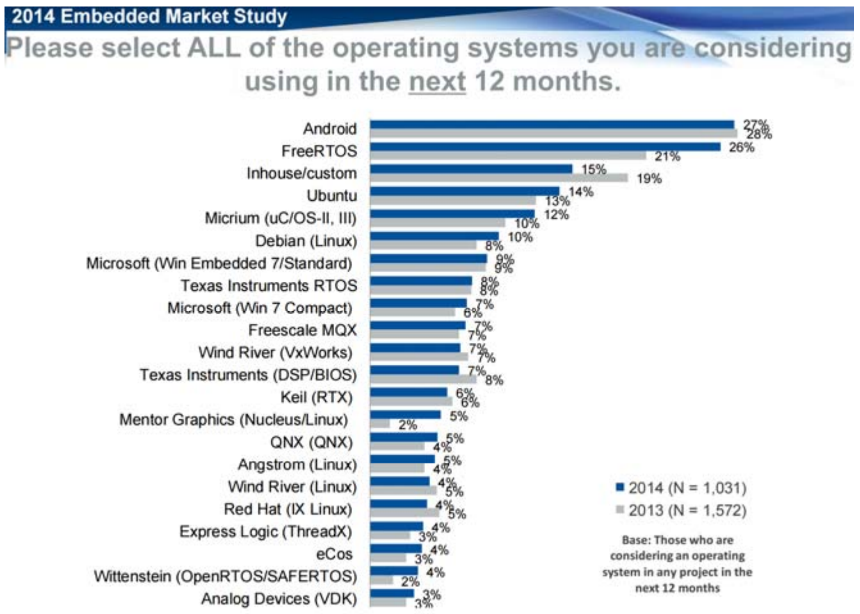
http://webpages.uncc.edu/~jmconrad/ECGR4101-2015-08/Notes/UBM%20Tech%202015%20Presentation%20of%20Embedded%20Markets%20Study%20World%20Day1.pdf

RTOS Selection Factors:



http://bd.eduweb.hhs.nl/es/2014-embedded-market-study-then-now-whats-next.pdf

RTOS Popularity



http://bd.eduweb.hhs.nl/es/2014-embedded-market-study-then-now-whats-next.pdf

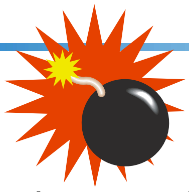
EMBEDDED SOFTWARE QUALITY CRISIS:

Avoid Betting Your
Company's Future on
Bad Software

Philip Koopman, Ph.D.

Edge Case Research LLC
Carnegie Mellon University

Overview



- **Step #1: Admit you have a problem**
  - Software problems are pervasive
  - Are you going to wait until you're on CNN to do something?
- **Act as if your company lives or dies by its software**
  - Software is a core competency, whether you like it or not
- **Business as usual isn't going to work**
  - Stakes are increasing, and software is getting more complex
  - Testing doesn't turn bad software into good software
- **Follow the path toward better software**
  - People, processes, comprehensive quality assurance

Step #1: Admit You Have A Problem

01010101010101010101010101010101010
10101010101010101010101010101010100
01010101010101010101010101010101010
1010101010101010111010101010101010100
01010101010101010101010101010101010
10101010101010101010101010101010100

3

One Software Mistake Is All It Takes

Edge Case Research

- Bad software can tarnish the brand…or kill the company

Knight Capital Says Trading Glitch Cost It $440 Million

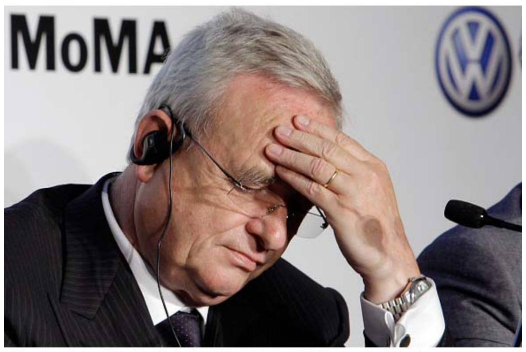
By NATHANIEL POPPER AUGUST 2, 2012 9:07 AM 356 Comments

Runaway Trades Spread Turmoil Across Wall St.

Errant trades from the Knight Capital Group began hitting the New York Stock Exchange almost as soon as the opening bell rang on Wednesday. Brendan McDermid/Reuters

The Knight Capital Group announced on Thursday that it lost $440 million when it sold all the stocks it accidentally bought Wednesday morning because a computer glitch.

http://dealbook.nytimes.com/2012/08/02/knight-capital-says-trading-mishap-cost-it-440-million/?_r=0



The scandal cost Martin Winterkorn his position as chief executive of VW Photo: AP/Richard Drew

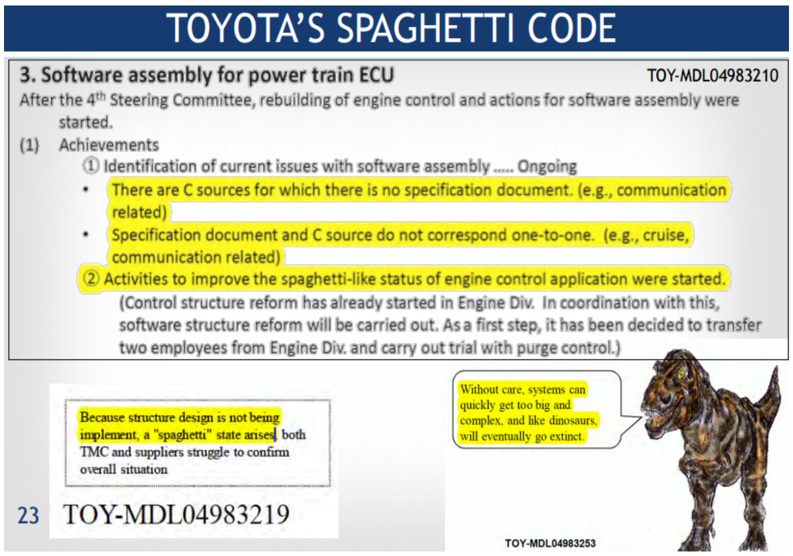
"Diesel-Gate"

http://www.telegraph.co.uk/finance/newsbysector/industry/engineering/12020564/Volkswagen-shares-to-double-as-dieselgate-crisis-abates-says-analyst.html

4

Some Code Seems Pervasively Bad

Edge Case Research



TOYOTA'S SPAGHETTI CODE

3. Software assembly for power train ECU TOY-MDL04983210

After the 4th Steering Committee, rebuilding of engine control and actions for software assembly were started.

(1) Achievements

① Identification of current issues with software assembly ..... Ongoing

- There are C sources for which there is no specification document. (e.g., communication related)
- Specification document and C source do not correspond one-to-one. (e.g., cruise, communication related)

② Activities to improve the spaghetti-like status of engine control application were started.

(Control structure reform has already started in Engine Div. In coordination with this, software structure reform will be carried out. As a first step, it has been decided to transfer two employees from Engine Div. and carry out trial with purge control.)

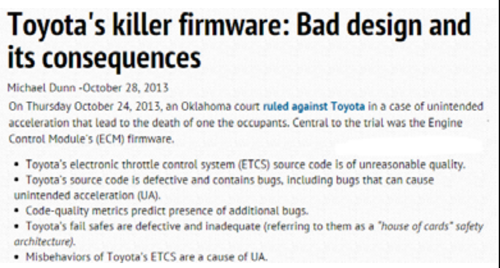
Because structure design is not being implement, a "spaghetti" state arises both TMC and suppliers struggle to confirm overall situation

23 TOY-MDL04983219

Without care, systems can quickly get too big and complex, and like dinosaurs, will eventually go extinct.

TOY-MDL04983253

http://www.safetyresearch.net/Library/BarrSlides_FINAL_SCRUBBED.pdf



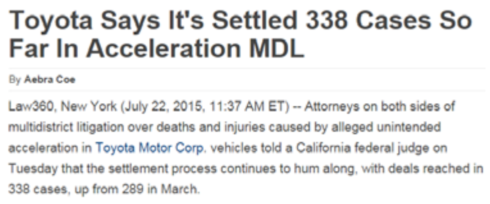
Toyota's killer firmware: Bad design and its consequences

Michael Dunn -October 28, 2013

On Thursday October 24, 2013, an Oklahoma court ruled against Toyota in a case of unintended acceleration that lead to the death of one the occupants. Central to the trial was the Engine Control Module's (ECM) firmware.

- Toyota's electronic throttle control system (ETCS) source code is of unreasonable quality.
- Toyota's source code is defective and contains bugs, including bugs that can cause unintended acceleration (UA).
- Code-quality metrics predict presence of additional bugs.
- Toyota's fail safes are defective and inadequate (referring to them as a *"house of cards" safety architecture).*
- Misbehaviors of Toyota's ETCS are a cause of UA.

http://www.edn.com/design/automotive/4423428/Toyota-s-killer-firmware--Bad-design-and-its-consequences



Toyota Says It's Settled 338 Cases So Far In Acceleration MDL

By Aebra Coe

Law360, New York (July 22, 2015, 11:37 AM ET) -- Attorneys on both sides of multidistrict litigation over deaths and injuries caused by alleged unintended acceleration in Toyota Motor Corp. vehicles told a California federal judge on Tuesday that the settlement process continues to hum along, with deals reached in 338 cases, up from 289 in March.

http://www.law360.com/articles/681915/toyota-says-it-s-settled-338-cases-so-far-in-acceleration-mdl

© 2016 Edge Case Research LLC 5

But It Only Takes One Bad Line of Code

Edge Case Research

- **This is the bad line of code for Heartbleed:**

memcpy(bp, pl, payload);

- Classic buffer overflow vulnerability
  - Copies "payload" bytes from pl to bp
  - Reads other user's data, including secret keys, if payload value is too big

How Heartbleed Works: The Code Behind the Internet's Security Nightmare

Eric Limer

Filed to: HEARTBLEED 4/09/14 2:59pm

http://gizmodo.com/how-heartbleed-works-the-code-behind-the-internets-se-1561341209

By now you've surely heard of Heartbleed, the hole in the internet's security that exposed countless encrypted transactions to any attacker who knew how to abuse it. But how did it actually work? Once you break it down, it's actually incredibly simple. And a little hilarious. But mostly terrifying.

© 2016 Edge Case Research LLC 6

There Are Too Many Examples

- And probably many more that aren't public

ANDY GREENBERG SECURITY 07.21.15 6:00 AM

HACKERS REMOTELY KILL A JEEP ON THE HIGHWAY—WITH ME IN IT

Miller attempts to rescue the Jeep after its brakes were remotely disabled, sending it into a ditch. ANDY GREENBERG/WIRED

http://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/

To keep a Boeing Dreamliner flying, reboot once every 248 days

by Edgar Alvarez | @abcdedgar | May 1st 2015 At 6:34pm

http://www.engadget.com/2015/05/01/boeing-787-dreamliner-software-bug/

1991: Patriot System Misses a Scud Missile Due To Software Defect: 28 Americans Dead

http://www.fas.org/spp/starwars/gao/im92026.htm
https://en.wikipedia.org/wiki/MIM-104_Patriot#/media/File:Patriot_missile_launch_b.jpg

7

How Bad Can It Possibly Be?

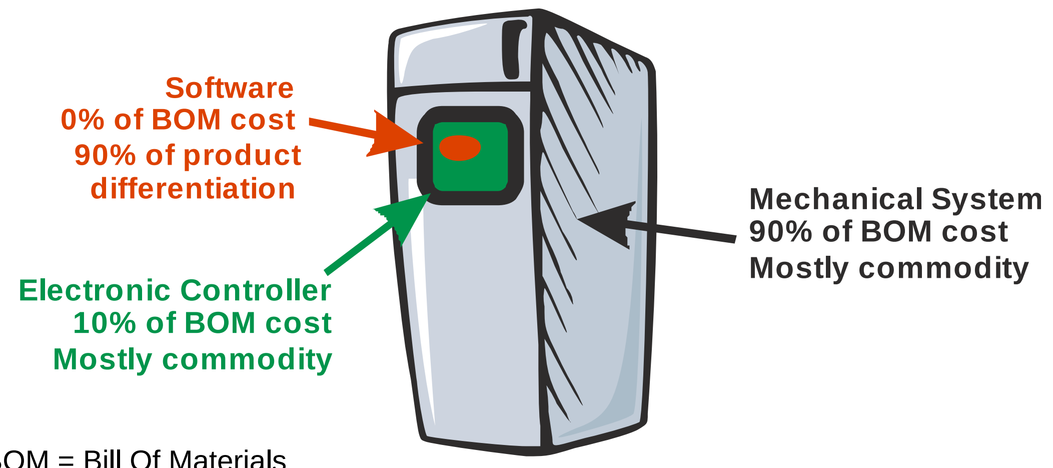
- For YOUR product, what is the worst possible outcome:
  - For a software bug?
    - People killed or injured?
    - Property damage?
    - Cost to deploy a fix?
    - Loss of brand reputation and customer confidence?

https://www.flickr.com/photos/sylvar/3119015160/in/photolist-5KBLxs-4xLkYf
CC BY 2.0 (cropped and contrast enhance)

  - For a malicious attack (assume an attack succeeds)?
  - Hint: *The answer is the same* for both bugs and attacks

8

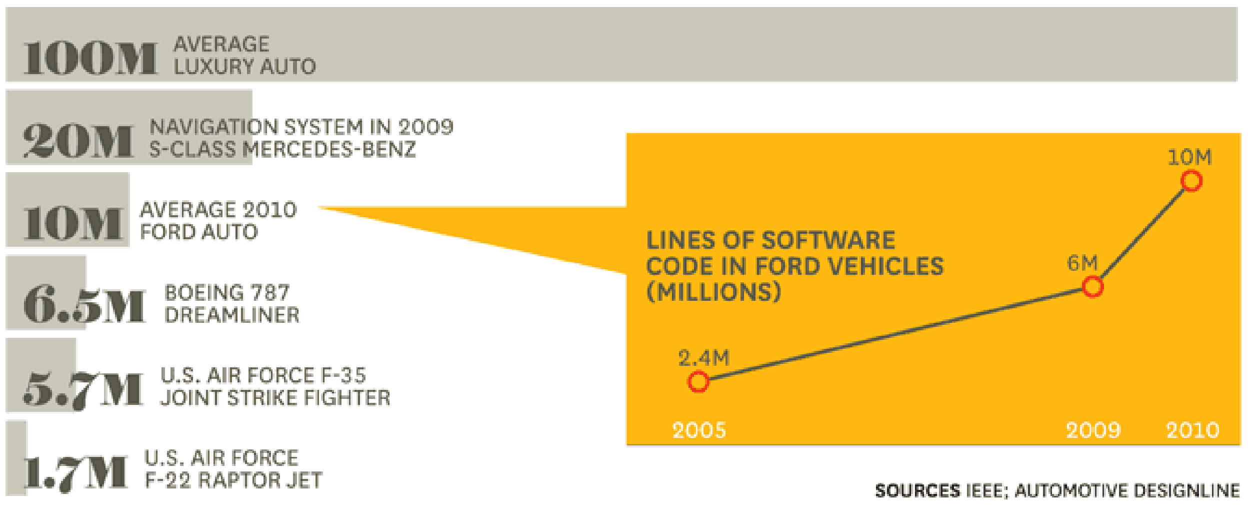
Act As If Your Products Live Or Die By Their Software



BOM = Bill Of Materials

Software Differentiates Products

- Software is a huge value-add. Take it seriously.



https://hbr.org/resources/images/article_assets/hbr/1006/F1006A_B_lg.gif

Large Scale Production = Big Problems

Edge Case Research

Toyota recalls 625k Prius models for faulty hybrid software

http://www.autoblog.com/2015/07/15/toyota-recalls-625k-prius-faulty-hybrid-software/

Honda, Yes Honda, Recalls 175,000 Cars For Unintended Acceleration

Patrick George 7/10/14

Bloomberg reports that all new hybrid Honda Fit subcompact and Vezel small crossover models sold in Japan since last will be recalled due to a software problem with the engine control system. They did not elaborate, but said the problem could lead to unintended acceleration.

http://jalopnik.com/honda-yes-honda-recalls-175-000-cars-for-unintended-a-1603215615

theguardian

Samsung keyboard bug leaves 600m Android devices exposed to hackers

Vulnerability remains months after discovery, allowing hackers to eavesdrop on calls, steal data and activate camera, microphone and GPS remotely

Samuel Gibbs
17 June 2015

Hundreds of millions of Samsung smartphones are vulnerable to hacking thanks to the built-in keyboard. Photograph: Samuel Gibbs for the Guardian

http://www.theguardian.com/technology/2015/jun/17/samsung-keyboard-bug-android-hack

© 2016 Edge Case Research LLC 11

Software Is A Core Competency

Edge Case Research

- **Embedded software is *not* free!**
  - 1,000 lines of code ➔ $40,000
  - 10,000 lines of code ➔ $400,000
  - 100,000 lines of code ➔ $4,000,000
  - 1,000,000 lines of code ➔ $40,000,000

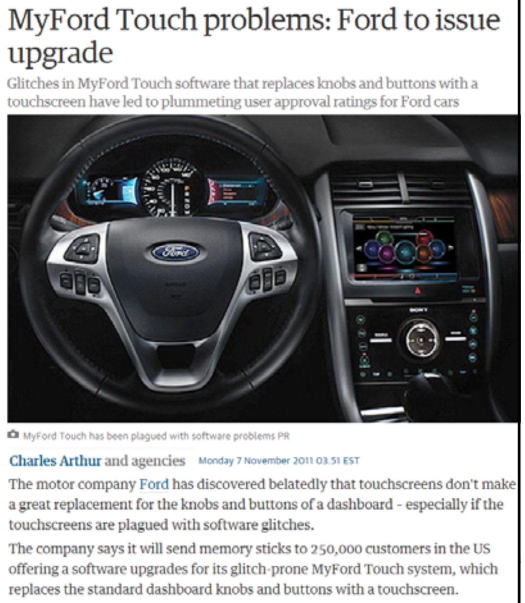
$15-$50 per line of source code

- **Big software needs to be managed as if it matters**
  - It provides differentiation
  - It's expensive to develop, and difficult to get right
  - It's even more expensive if a big field failure happens
  - It should be a first class citizen in skills and project management

© 2016 Edge Case Research LLC 12

Embedded Software Is Difficult

- **Customers expect embedded SW to be essentially perfect**
  - Upgrades can be painful to deploy
  - Bugs can lead to class action lawsuits
- **Severe technical challenges**
  - Limited hardware resources
  - Real-time operation
  - Interaction with system-specific sensors and actuators


MyFord Touch problems: Ford to issue upgrade

Glitches in MyFord Touch software that replaces knobs and buttons with a touchscreen have led to plummeting user approval ratings for Ford cars

MyFord Touch has been plagued with software problems PR

Charles Arthur and agencies Monday 7 November 2011 03.51 EST

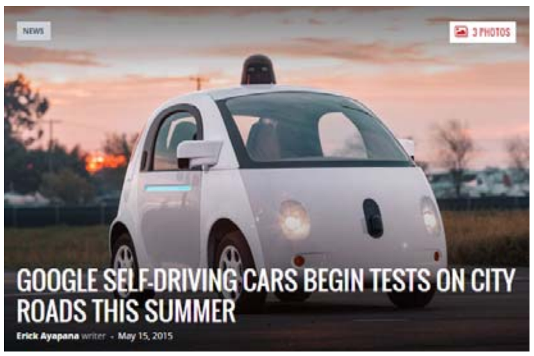
The motor company Ford has discovered belatedly that touchscreens don't make a great replacement for the knobs and buttons of a dashboard - especially if the touchscreens are plagued with software glitches.

The company says it will send memory sticks to 250,000 customers in the US offering a software upgrades for its glitch-prone MyFord Touch system, which replaces the standard dashboard knobs and buttons with a touchscreen.

http://www.theguardian.com/technology/2011/nov/07/myford-touch-problems-upgrade

Safety Concerns Are Increasing

- **Uncontrolled release of energy is a safety issue**
  - As products have more control authority, they control more energy
    - – Release of energy directly (e.g., electricity from batteries)
    - – Control of mechanical systems (e.g., control of combustion engine)
  - If your system turned on all its actuators full-force, what happens?
    - – How do you know there is no bug that causes this to happen?
- **Regulation likely to increase**
  - IEC 60730 safety standard required for European appliances


NEWS
3 PHOTOS
GOOGLE SELF-DRIVING CARS BEGIN TESTS ON CITY ROADS THIS SUMMER
Erick Ayapana writer - May 15, 2015

http://www.motortrend.com/news/google-self-driving-cars-begin-tests-on-city-roads-this-summer/

Embedded Security Is A Huge Deal

- **Everything on the Internet is being attacked 24x7**
  - Security in every industry will soon get its 15 minutes of fame

HACKER CAN SEND FATAL DOSE TO HOSPITAL DRUG PUMPS

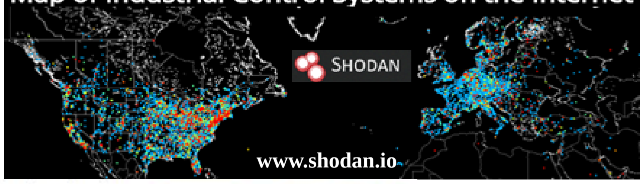
KIM ZETTER SECURITY 06.08.15 7:00 AM

Hospira's drug infusion pumps include a serial cable (the wide grayish-white cable with the single red stripe on one edge) that connects the communications module to the main pump board. BILLY RIOS

administer. Because the libraries don't require authentication, Rios found that anyone on the hospital's network—including patients in the hospital or a hacker accessing the pumps over the Internet—can load a new drug library that alters the limits for a drug.

http://www.wired.com/2015/06/hackers-can-send-fatal-doses-hospital-drug-pumps/

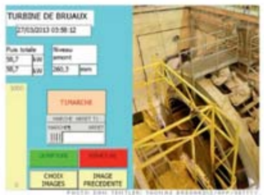
Map of Industrial Control Systems on the Internet



A hydroelectric plant

French electric companies apparently like to put their hydroelectric plants online. Tentler found three of them using Shodan.

This one has a big fat button that lets you shut off a turbine. But what's 58,700 Watts between friends, right?



Wait, does that say kilowatts?

It's not just France that has a problem. The U.S. Department of Homeland Security commissioned researchers last year to see if they could find industrial control systems for nuclear power plants using Shodan. They found several.

Tentler told DHS about all the power plants he found -- actually, DHS called him after he accessed one of their control systems.

"a big fat button lets you shut off a turbine"

(No login credentials required)

http://money.cnn.com/gallery/technology/security/2013/05/01/shodan-most-dangerous-internet-searches/4.html

© 2016 Edge Case Research LLC **15**

Security Matters for Industrial Systems!

- **Personal info theft isn't the only security issue:**

Hack attack causes 'massive damage' at steel works

The hack attack led to failures in plant equipment and forced the fast shut down of a furnace

A blast furnace at a German steel mill suffered "massive damage" following a cyber attack on the plant's network, says a report.

http://www.bbc.com/news/technology-30575104

Hackers caused power cut in western Ukraine

12 January 2016

Ukraine has been forced to turn to back-up power sources in recent months following a spate of power cuts

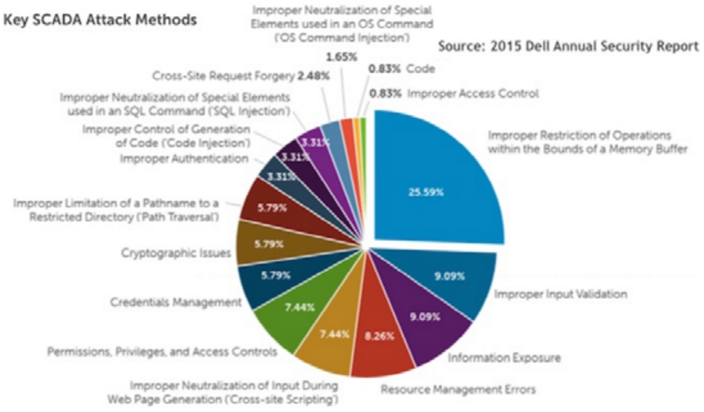
A power cut in western Ukraine last month was caused by a type of hacking known as "spear-phishing", says the US Department of Homeland Security (DHS).

http://www.bbc.com/news/technology-35297464

Attacks Against SCADA Systems Doubled in 2014: Dell

By Mike Lennon on April 13, 2015

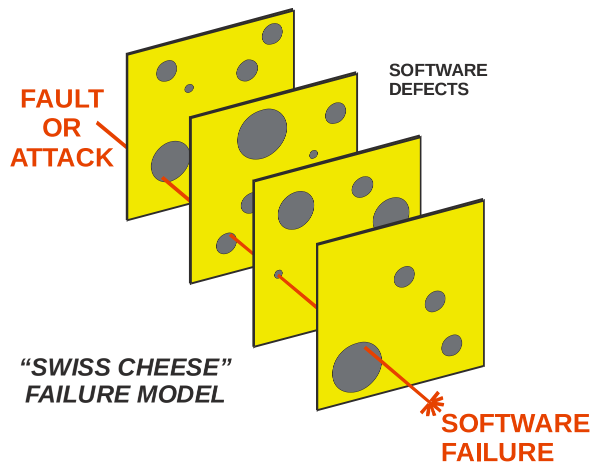
Dell SonicWALL saw global SCADA attacks increase against its customer base from 91,676 in January 2012 to 163,228 in January 2013, and 675,186 in January 2014.



http://www.securityweek.com/attacks-against-scada-systems-doubled-2014-dell

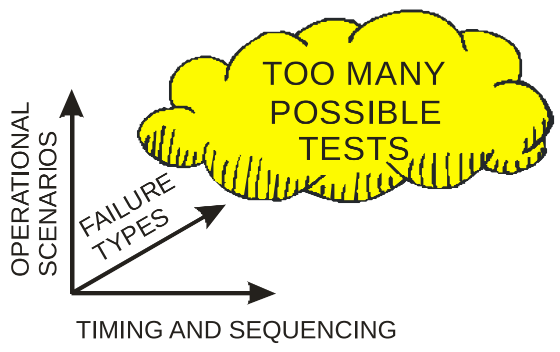
© 2016 Edge Case Research LLC **16**

Business As Usual Isn't Going To Work



Product Testing Won't Find All Bugs

- **Testing bad software simply makes it less bad**
  - Testing cannot produce good software all on its own



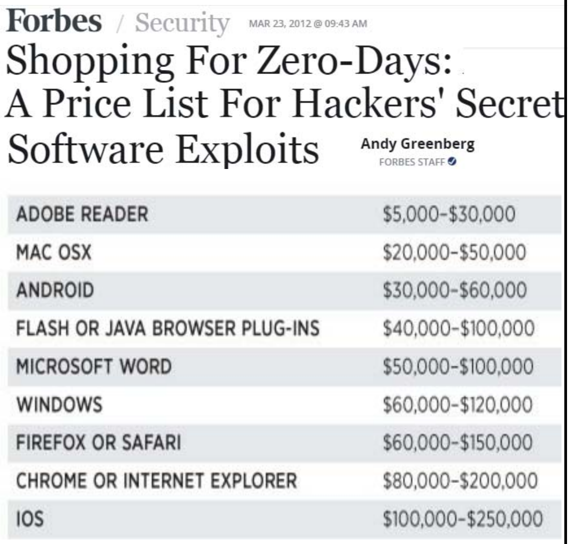
- **One third of faults take more than 5000 years to manifest**

Adams, N.E., "Optimizing preventive service of software product," IBM Journal of Research and Development, 28(1), p. 2-14, 1984. (Table 2, pg. 9, 60 kmonth column)

- Do you test for more than 5000 years of use?
- Your customers will regularly experience bugs that you will not see during testing

Security Testing Isn't Enough

- **Security testing, at best, finds currently known problems**
  - Some problems known but not publicly announced
  - More problems will be discovered after you ship
- **Attacks will likely increase over time**
  - How will you respond to
  - emergent threats?



Forbes / Security MAR 23, 2012 @ 09:43 AM

Shopping For Zero-Days: A Price List For Hackers' Secret Software Exploits

Andy Greenberg FORBES STAFF

ADOBE READER	\$5,000-\$30,000
MAC OSX	\$20,000-\$50,000
ANDROID	\$30,000-\$60,000
FLASH OR JAVA BROWSER PLUG-INS	\$40,000-\$100,000
MICROSOFT WORD	\$50,000-\$100,000
WINDOWS	\$60,000-\$120,000
FIREFOX OR SAFARI	\$60,000-\$150,000
CHROME OR INTERNET EXPLORER	\$80,000-\$200,000
IOS	\$100,000-\$250,000

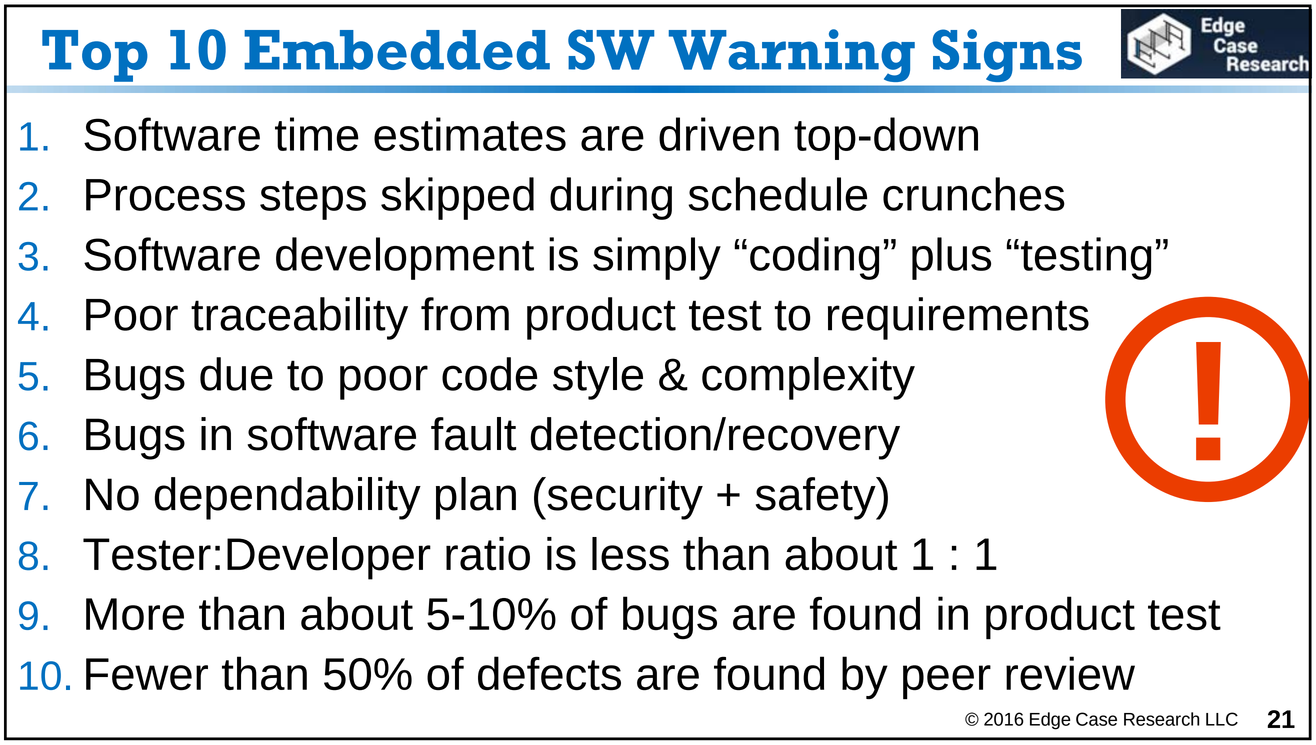
http://www.forbes.com/sites/andygreenberg/2012/03/23/shopping-for-zero-days-an-price-list-for-hackers-secret-software-exploits/

© 2016 Edge Case Research LLC 19

You Can't Fix Bad Software

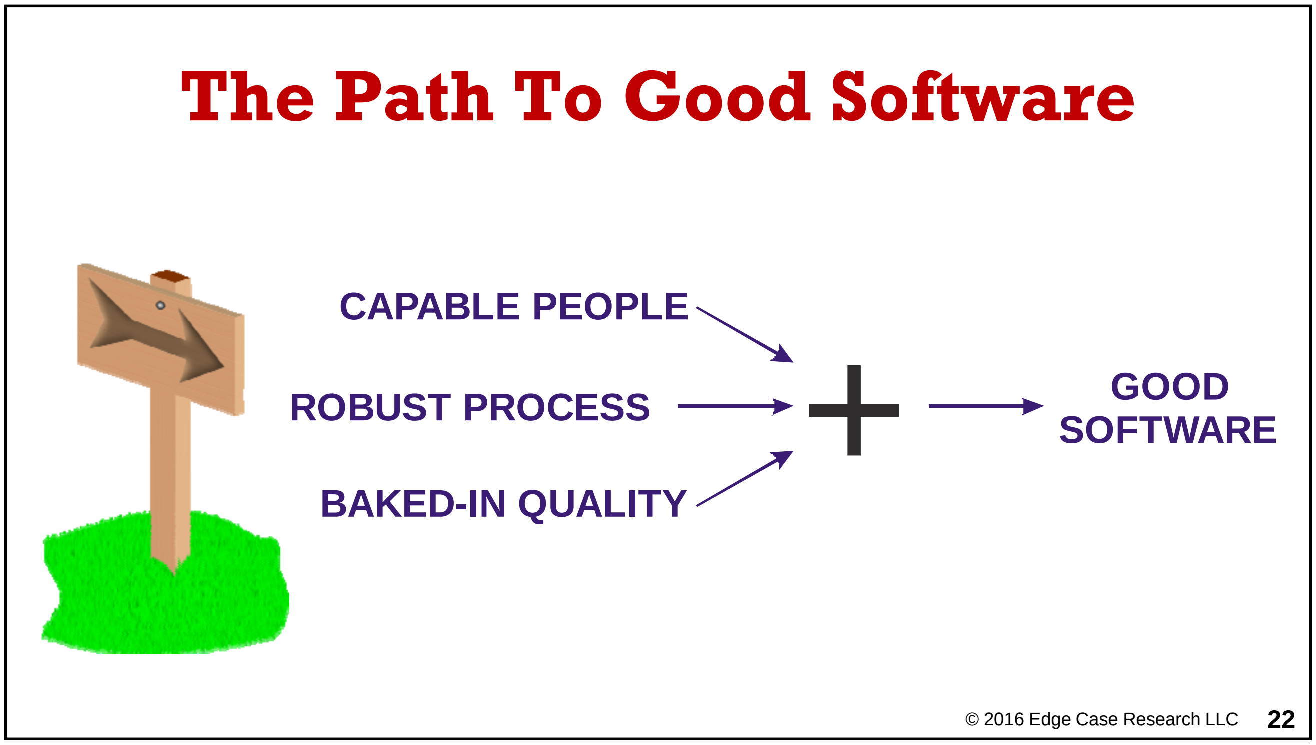
- **You can't test in quality, safety, or security**
- **The cheapest, smartest way to fix bad software is to throw it away and start over**
  - There's never time to do it right…
    - And usually it seems like there's no time to do it over
  - But … incremental improvement strategies can work
    - Requires cultural change
    - Requires commitment to good software at all levels of organization
    - Commitment must survive a "but we have to ship next week" crisis

© 2016 Edge Case Research LLC 20



Top 10 Embedded SW Warning Signs

1. Software time estimates are driven top-down
2. Process steps skipped during schedule crunches
3. Software development is simply “coding” plus “testing”
4. Poor traceability from product test to requirements
5. Bugs due to poor code style & complexity
6. Bugs in software fault detection/recovery
7. No dependability plan (security + safety)
8. Tester:Developer ratio is less than about 1 : 1
9. More than about 5-10% of bugs are found in product test
10. Fewer than 50% of defects are found by peer review



The Path To Good Software

$$\begin{tikzcd}
\text{CAPABLE PEOPLE} \arrow[dr] & & \\
\text{ROBUST PROCESS} \arrow[r] & + \arrow[r] & \text{GOOD SOFTWARE} \\
\text{BAKED-IN QUALITY} \arrow[ur] & &
\end{tikzcd}$$

Requires a Multi-Prong Approach

- **People**
  - Good skills; trained on process and technical skills
  - Full time software professionals, regardless of formal education
- **Process**
  - Robust, methodical, well defined engineering process
  - Good technical practices, especially embedded-specific issues
  - Project management that appreciates the cost + benefit of software
- **Quality**
  - Checks and balances to ensure quality is baked in
  - Emphasizes good engineering, not just testing

Next Steps

- **Skills check-up**
  - Do your people have the skills and tools they need?
  - Look at a recent project and see how things look
- **Process evaluation and improvement**
  - Is your process the right level of rigor and formality?
  - Look at your process and see if there are big gaps
- **Software Quality Health Check**
  - Are your checks and balances in place and working?
  - Look at your bug track record and see if it looks OK