Embedded Systems Methodologies

Presented to: IEEE Information Technology Professionals Conference

March 16, 2018

The issues (not my words)

- Nancy Leveson, a professor of aeronautics and astronautics at the Massachusetts Institute of Technology: "*The problem is that we are attempting to build systems that are beyond our ability to intellectually manage.*"
- Dutch computer scientist Edsger Dijkstra wrote in 1988: "*The programmer has to be able to think in terms of conceptual hierarchies that are much deeper than a single mind ever needed to face before.*"
- Michael Barr, an expert witness for the plaintiff in the Toyota case: "*You have software watching the software. If the software malfunctions and the same program or same app that is crashed is supposed to save the day, it can't save the day because it is not working.*"
- Chris Granger, a software developer who worked as a lead at Microsoft on Visual Studio: "*Visual Studio is one of the single largest pieces of software in the world. It's over 55 million lines of code…and more than 98 percent of it is completely irrelevant*"

Reference: [CSA]

Motivation

- Embedded Systems commonly have low development overhead, minimal memory or storage per unit, and are heavily cost-driven
  - There are significant market pressures to “shoot the Engineer and put it into production” and not heavily invest in design rigor

- The Internet of Things (IoT) has proved that while capabilities can be delivered inexpensively
  - Releasing products that have Safety or Security vulnerabilities can cost hundreds of thousands of dollars in rework or MILLIONS in lawsuits

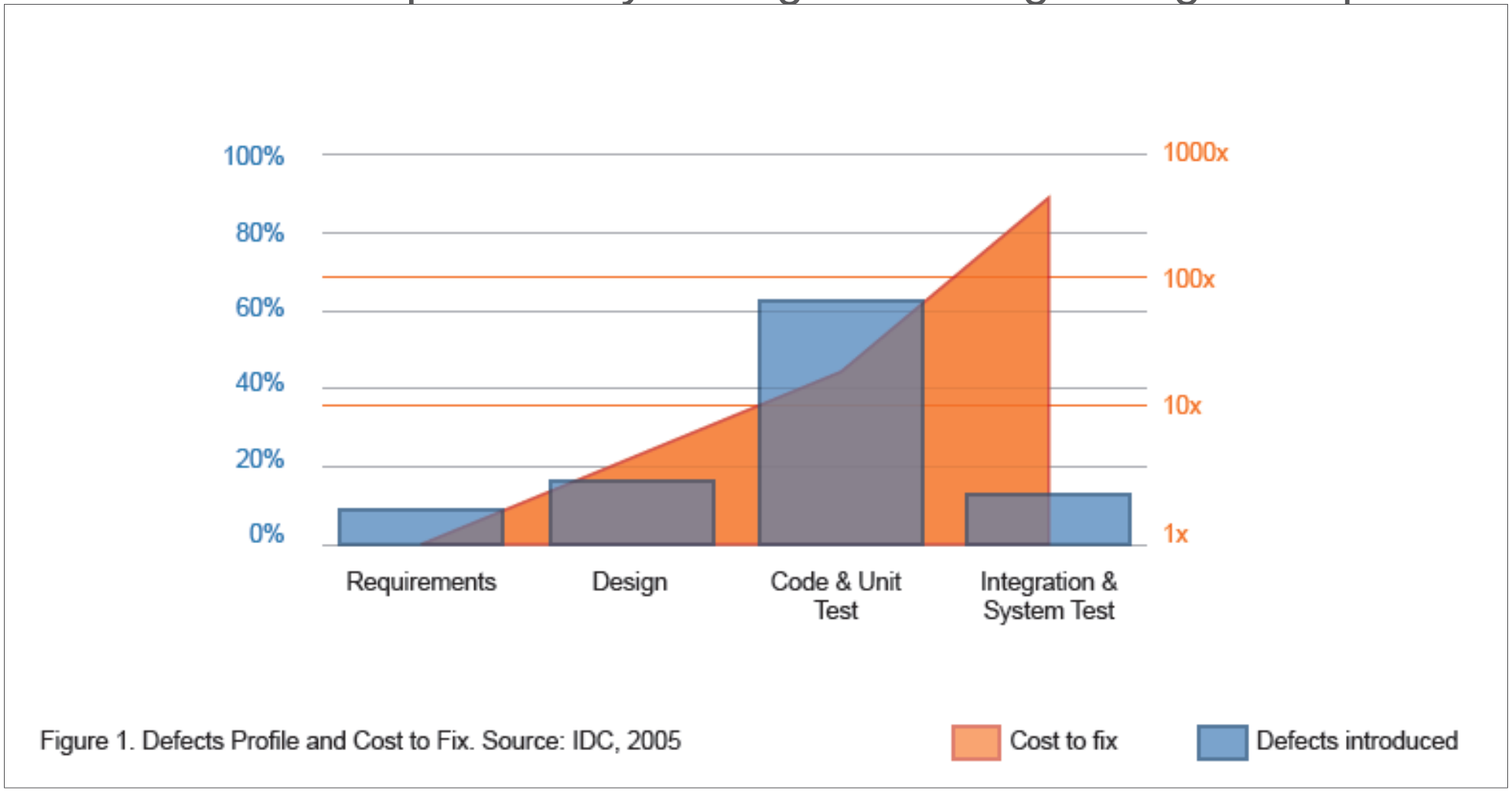
The “Playing Field”

- **Programming Research**, in “Addressing Security Vulnerabilities in Embedded Applications…” opined that **SECURITY IS JUST NOT A PRIORITY**
  - *“Most development organizations, perhaps unknowingly, subscribe to the iron triangle adage – ‘get to market fast, with all the features planned, and a high level of quality…pick two.’ And while quality has been part of the conversation, security is typically omitted.”* [PR1]

- Jay Thomas, in the **Embedded** magazine article “Software Standards 101: Tracing Code to Requirements,” opined that it is an **industry standard** that making systems safe or secure includes ten steps (see slide 9) [JT]

The “Playing Field” (cont.)

- Cost to fix increases exponentially through the design/integration process [PR2]



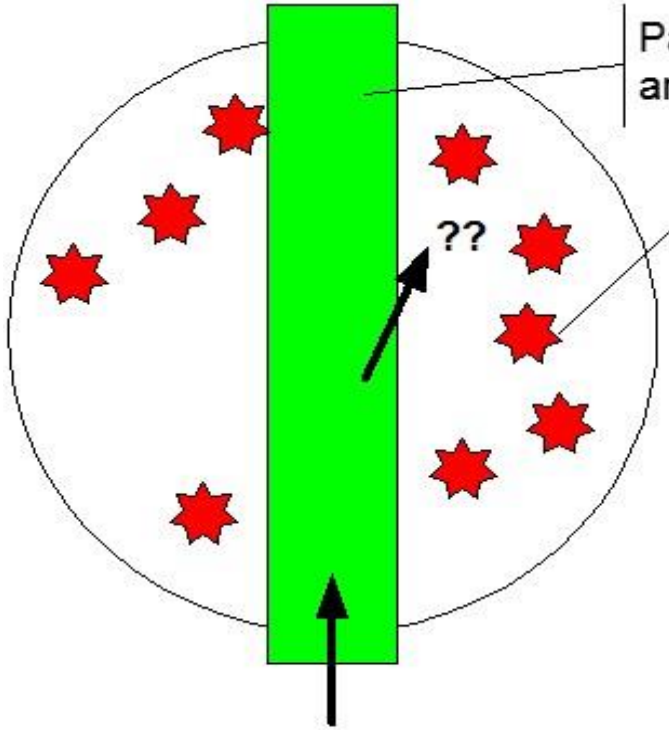
The “Playing Field” (cont.)

- The Mirai botnet attack leads us to believe that we need to take the risk seriously
  - IoT companies especially need to evaluate any liability incurred due to Security flaws in their products
  - Even Operating Systems may not be secure
  - According to a recent survey 38% of JAVA applications use a package with a known flaw (aopalliance-1.0.jar) [SoSS]

- Design Complexity
  - While speaking at the **EE Live! 2014** conference, I was struck by the number of presentations whose premise was, “Designs are complex. We need to add more structure to the design process to ensure success!”
  - I was pleasantly surprised, since I had been invited by the Conference Director to present some relevant SE concepts to answer this need
  - At the risk of “being the hammer that thinks everything is a nail,” I do think Design Rigor can benefit the safety and security of embedded systems

The “Minefield”

- It sounds like the “ten steps” mentioned above might be a good place to start
  - Engility’s illustration shows that DEFECTS act like mines in a minefield
  - “Code and Test” methodologies just CLEAR A PATH through the minefield
  - System overload, operator error, or race conditions could force the system into unexplored territory



Path cleared by "code and fix" processes

Application "mine field"

Conditions off the path include:

- Overload
- Configuration
- Resource contention
- Data error
- Operator error
- Hardware faliure
- Race conditions

“Design 2.0:” Improving the Process

Jay Thomas' "10 Steps" of Design Rigor [PR1]

- **"Perform a safety or security assessment,**
- **Determine a target system failure rate,**
- **Use the target system failure rate to determine the appropriate level of development rigor,**
- **Use a formal requirements capture process,**
- **Create software that adheres to an appropriate coding standard,**
- **Trace all code back to their source requirements,**
- **Develop all software and system test cases based on requirements,**
- **Trace test cases to requirements,**
- **Use coverage analysis to test completeness against both requirements and code,**
- **For certification, collect and collate the process artifacts required to demonstrate that an appropriate level of rigor has been maintained."** [JT]

Since Embedded systems have low development overhead, minimal memory or storage per unit, and are heavily cost-driven, these comments are encouraging to those who support more rigorous design.

What Should the Design Process Look Like???

1. User Needs (What do I really want to Know?) – concept of operations
2. Requirements (derived from User Needs)
3. Design (allocate requirements to architecture elements)
4. Sell off the requirements (V&V)
    1. Verification (Did we build the system right?)
    2. Validation (Did we build the right system?)
5. Integration/Certification

According to Jon Friedman, Aerospace and Defense Industry Marketing Manager at **MathWorks** in Natick, MA, "*A well-defined process includes the following phases: requirements capture and validation, design and test, implementation and integration, final verification, and sign-off*." [MAE]

Compare to the “10 Steps”

Number	Step	Design 2.0 Process
1	Perform a safety or security assessment	User Needs
2	Determine a target system failure rate	User Needs
3	Use the target system failure rate to determine the appropriate level of development rigor	User Needs
4	Use a formal requirements capture process	Requirements
5	Create software that adheres to an appropriate coding standard,	Design
6	Trace all code back to their source requirements	Design
7	Develop all software and system test cases based on requirements,	Verification & Validation
8	Trace test cases to requirements,	Verification & Validation
9	Use coverage analysis to test completeness against both requirements and code	Verification & Validation
10	For certification, collect and collate the process artifacts required to demonstrate that an appropriate level of rigor has been maintained	Certification

A Few Words About Requirements

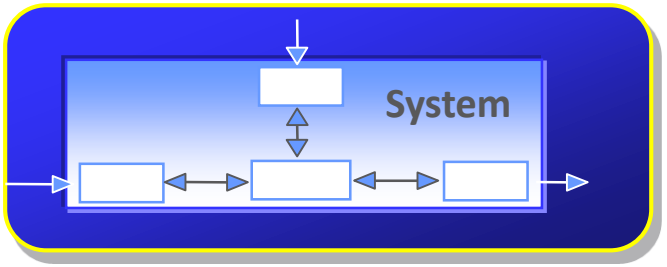
- **Requirements are**:
  - **Specific**, **quantifiable**, and **verifiable** statements that drive the development or maintenance of capabilities to **meet stakeholder needs**.
  - Requirements define a **capability** in terms of its **function**, **interfaces**, **performance**, and **constraints**, taking into account the user's preferences and expectations

Natural Language Processing (NLP)

- NLP analysis of requirements was pioneered by NASA researchers Wilson, Rosenberg, and Hyatt in the 1990s [MAE]
- NASA identified "Quality Attributes" that requirements should possess:
  - "***Complete*** *- precisely defines the system's responses to all real-world situations the system will encounter.*
  - ***Consistent*** *- does not contain conflicts between requirements statements.*
  - ***Correct*** *- accurately identifies the conditions of all situations the system will encounter and precisely defines the system's response to them.*
  - ***Modifiable*** *- as a logical structuring with related concerns grouped together.*
  - ***Ranked*** *- organizes the specification statements by importance and/or stability (which may conflict with the document's modifiability).*
  - ***Traceable*** *- identifies each requirement uniquely.*
  - ***Unambiguous*** *- states all requirements in such a manner that each can only be interpreted one way.*
  - ***Valid*** *- all project participants can understand, analyze, accept or approve it.*
  - ***Verifiable*** *- must be consistent with related specifications at other (higher and lower) levels of abstraction*" [QRA]
- Companies such as QRA Corp. provide NLP products to identify requirements problems

Five Types of Requirements

1. Functional – *"What must it do?"*
2. Performance – *"How well must it do it?"*
3. Interface – "*What it must connect with?*
4. Constraint – *"What limitations and constraints does the customer want?"*
5. Environment – *"Under what conditions must it operate?"*



Functional

Performance

An Example of a Requirements Problem

- **System "Nexus" board**
  - Systems Engineering specified one circuit card to supply:
    - Several disparate interfaces (2 bidirectional parallel, 4 RS-232 ports)
    - Needed to be "as fast as possible"
    - **Critical questions about performance requirements went unanswered**
  - Lead Engineer used "Code and Fix" approach ("mine field")
    - Circuit card design was over a year late
    - $250K cost growth due to additional personnel, prototypes, board layouts
  - Final post mortem revealed these parallel port test results:
    - COTS FIFO board – 2 msec per message transfer
    - "Nexus" board CPU only – 500 usec per message transfer
    - "Nexus" board DMA accel. – 128 usec per message transfer
    - **Destination system context switch was 20 msec MINIMUM!!!**

Requirement should have been about 10 msec per message transfer

Improving Code Quality

Design Rigor Supports a Secure System (the problem)

- **Programming Research**, in "How IoT is Making Security Imperative for all Embedded Software," recommended that software developers take more care in releasing new IoT products
  - *"Security problems often stem from the need to accelerate development and bring new products to market ahead of the competition"*
  - *"A majority of security vulnerabilities are a result of coding errors that go undetected in the development stage"*
  - *"Carnegie Mellon's Computer Emergency Response Team (CERT) found that 64% of vulnerabilities in the CERT National Vulnerability Database were the result of programming errors"* [PR0]

Design Rigor Supports a Secure System (the solution)

- **[IBID]** stated that organizations should incorporate coding standards (CERT C) and utilize the CERT-developed Common Weakness Environment (CWE) database [PR0]
  - MISRA-C and ISO/IEC are other available standards
  - Programming Research, Critical Software, or Jama Software offer static code analysis tools to assist with analysis of code to these standards
  - **(IBID)** *"An increasing number of organizations are making adherence to these guidelines and standards a requirement for both internal development organizations and outsourced application development vendors"* [PR0]

Cyber Security

- Massive data breaches at Target, OPM, and the Ukrainian Power System lend credence to the concern that vulnerabilities in our code need to be mitigated

- Chris Young, CEO, McAfee, in the Lawfare Podcast, "Chris Young on Cybersecurity and a Debate on Using Data to Protect Data," stated, "…*you have a massive change in the attack landscape … Ten years ago we saw about 25 new threats or new variants of malware a day … Today, about ten years later, we see over 500,000 new threats a day.*"[HAS]

- *"In late 2016, Dr. Johannes Ullrich at the Internet Storm Center conducted a simple experiment. [He] connected a DVR with a common user name and password to an ordinary cable modem and captured all packets going in and out of the DVR…The first attempt to log in to his DVR occurred in the first minute. Within the first hour, there were 54 unique attempts, approximately one per minute*." [HAS]

How to Avoid the “Minefield”

- The solution is clearly multi-layered
  - The **National Security Agency** has even made lightweight cryptologics available for public use in order to secure data in motion or at rest [S&S]
  - Does your operating system manage all processes equally, or can higher-priority applications be isolated and continue to operate if others crash (least privilege principle)? [HAS]
  - Can you leverage virtualization to partition apps to run on separate virtual machines? [HAS]

ADA, Qualified RTOS, etc.

- **Military and Aerospace Electronics**, in the article, “Safety and Security in Critical Avionics,” mentioned that the ADA programming language is still heavily used in avionics due to its tighter controls on Classes and Types [MAE]

- A **qualified** RTOS such as Green Hills Software’s **Integrity-178B** and DDC-I’s **DEOS** have been utilized in enough applications to prove their worth [MAE]

Additionally, Peer Reviews Find Defects

- The defects vary with the review object
  - **Requirements**: Is the system buildable? Is it testable?
  - **Design**: Does it meet its requirements?
  - **Implementation**: Does it correctly render the design? Is it maintainable?

- Your peers help find the defects
  - The people best equipped to help identify defects are other **technical** contributors
  - Managers do not participate

Peer Reviews Clear Defects Systematically

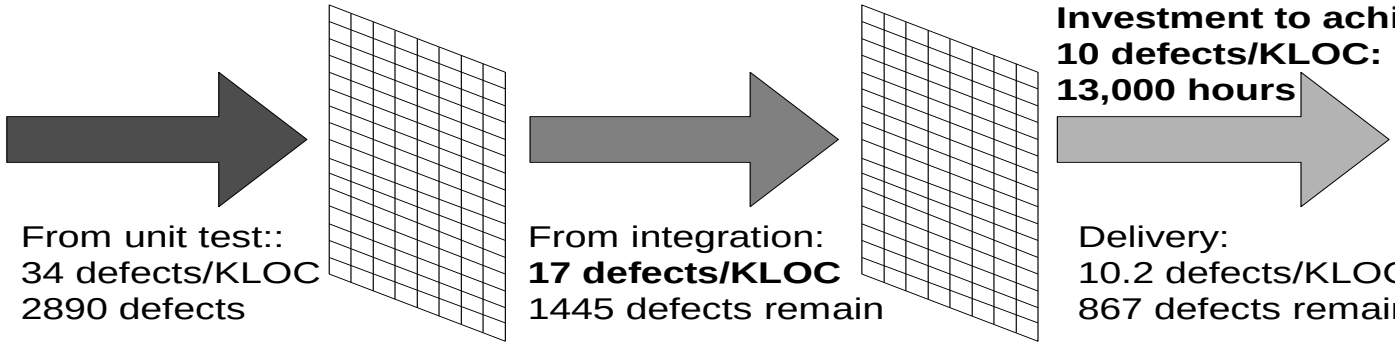
- When you do a peer review, you are not:
  - Mentally executing code
  - Proofreading documents

- Rather, you are:
  - Focused on the product's "criticals to quality"
  - Clearing the entire product of known defect types that affect those "criticals to quality"
  - As a **Technical Contributor**, you know what these are

- Highly recommend a checklist driven review as the method, common issues are [Wind]:
  - Buffer overflow
  - Parameters check (or lack thereof)
  - Logic errors
  - Redundancy management
  - Interaction accidents

Benefits of Peer Reviews

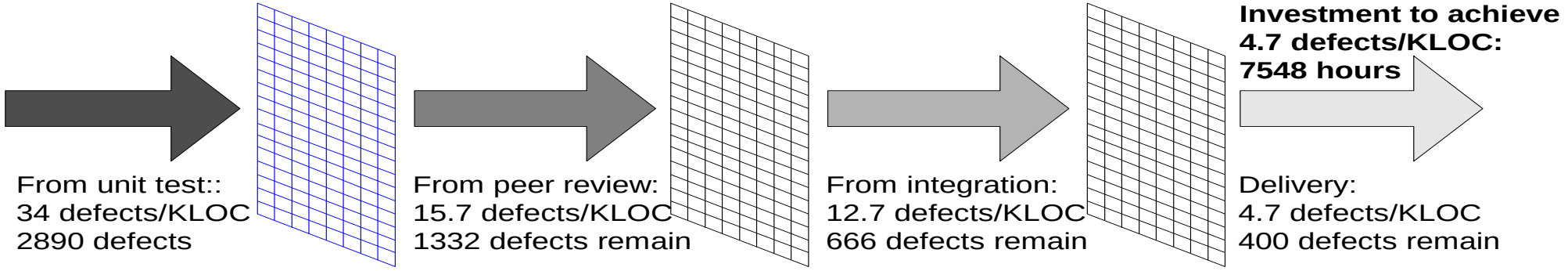
For an 85 KLOC project:



Integration test:
50% defect removal rate
5 hours/defect
7225 hours

System test
40% defect removal rate
10 hours/defect
5780 hours

Source: internal Engility Software Development Project

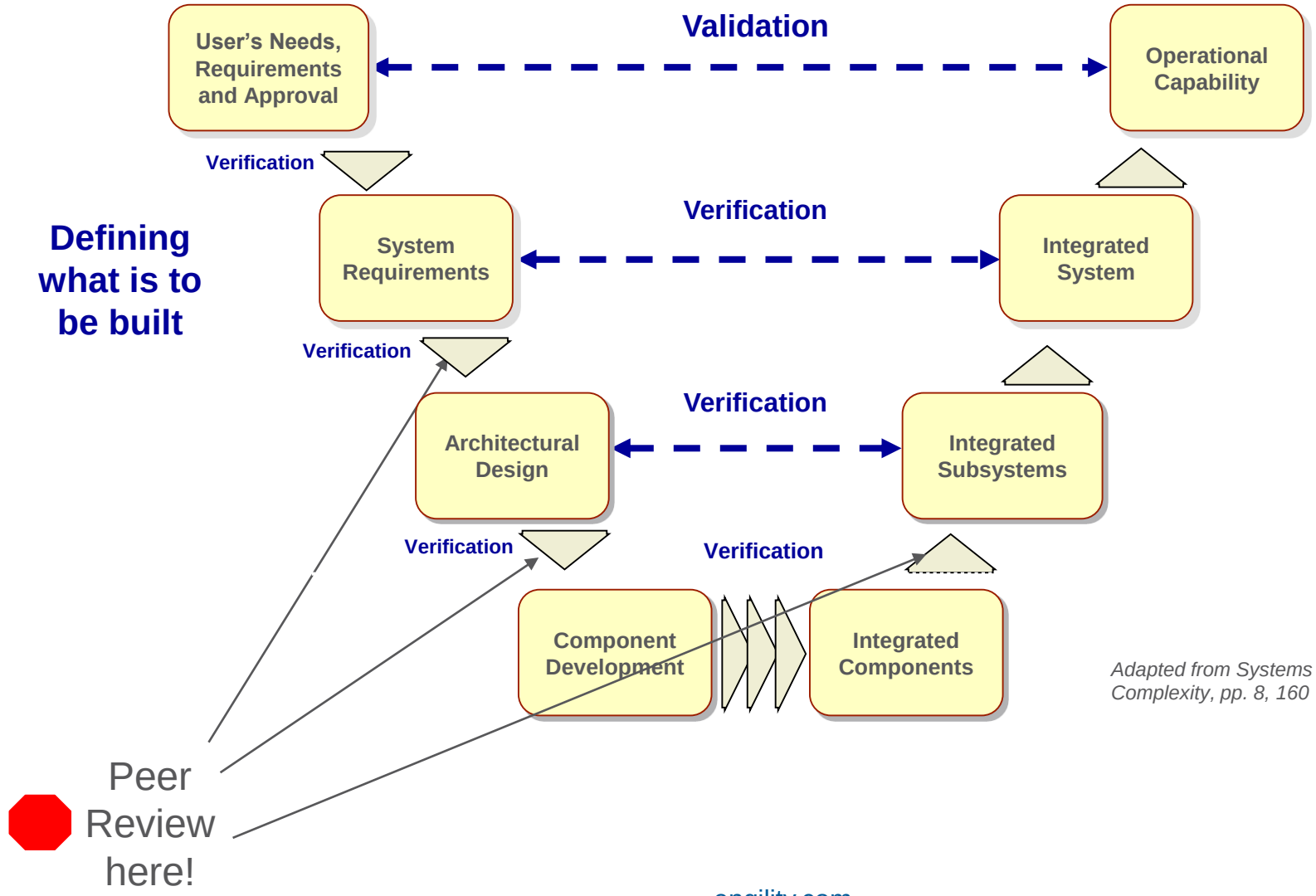


Peer review (**54% defect removal rate**)
611 LOC/review
11.2 (nominal) hours/peer review
1 (nominal) hour/defect (1-3)
1558 hours

Integration test:
50% defect removal rate
5 hours/defect
3330 hours

System test
40% defect removal rate
10 hours/defect
2660 hours

Some Peer Review Points



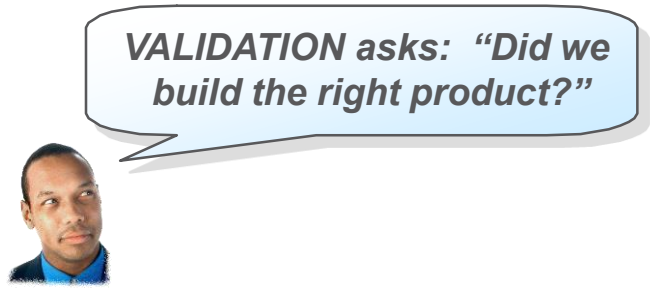
Adapted from Systems Engineering, Coping with Complexity, pp. 8, 160

Testing

What are Verification and Validation (V&V)?

- Processes to deliver a quality product that
  - Meets its specification
  - Satisfies the customers' needs
  - Works as intended in the operational environment
- Verification ensures the product correctly implements stated requirements

- Validation ensures the product is traceable to stated customer needs



Products and Requirements

- A product is anything that can be verified and validated
  - Single elements: requirements, design, software, hardware, or documentation
  - Single elements may be integrated together to become components within a larger product: a single stand-alone system or segments of a larger system
- Products are defined by their allocated requirements
- Products are verified and validated by evaluating their requirements against specified acceptance criteria
- Defects are problems that, if not corrected, would cause a product to not meet its acceptance criteria
- In the V&V world, “test” can be a
  - Verb referring to the verification activity – “test a product…”
  - Noun denoting a verification method – inspection, analysis, demo, test

Verification and Validation (V&V)

- May be performed by:
  - The development team building a product
  - Independent V&V teams within an organization

- V&V Methods – choose the most efficient:
  - **Inspection** is used for qualitative requirements that can be visually verified by sight, touch, or hearing.
  - **Analysis** is used when the requirement cannot be verified using another method or to augment another method
  - **Demonstration** is used where only **functionality** is specified by the requirement
  - **Test** is used when there is a **quantitative** aspect to the requirement

Flight Qualification

- NASA (DoD, FAA, etc.) sets standards for requirements traceability

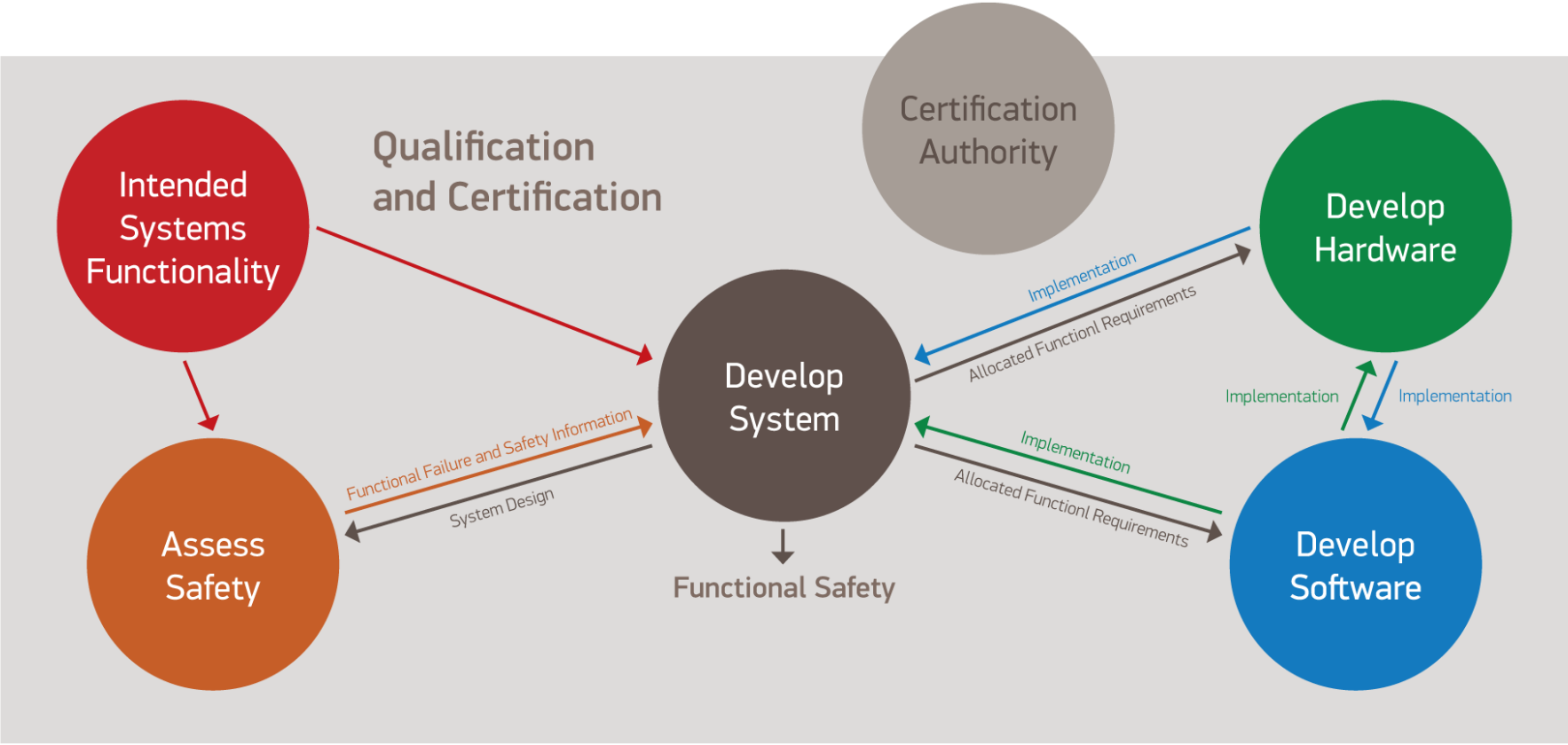
- **Jama S/W**, in their "Traceability Best Practices" white paper, recommends the following best practices [Jama]:
  - Trace relationships to represent systematic decomposition and test coverage
  - Ensure traceability reporting and proper coverage…
  - Assess the impact of change before it occurs…
  - Document changes for complete visibility and a detailed audit trail…
  - Stay synced…by referencing people and items…

Flight Certification

- Automated V&V: **Critical Software**, in the whitepaper, “Automated Verification and Validation,” stated, “*After test procedures are produced, all necessary test execution campaigns can be performed automatically when requested by the Test Engineers:*
  - *Automatic execution of all tests to be formally used for the certification record,*
  - *Automatic production of a report to be used as validation evidence.”* [A&A]

- **Critical Software**, in the whitepaper, “Safety Critical Validation,” stated that “[Reliability, Availability, Maintainability, and Safety] *RAMS analysis…provides useful inputs for requirements completeness and coherence, especially regarding safety and fail-safe issues.*”[SCV]
  - [next slide]

RAMS Diagram



Load Testing

- It is important to stress test any system to ensure performance is acceptable and repeatable
- Several types of testing are suggested: [API]
  - Spike testing - what happens when you hit that max stress level quite suddenly
  - Configuration testing - helps you find any changes to the pieces of your system that affect behavior or performance
  - Endurance testing - monitors continuous load, red-flagging any slow leaks that may be slowing you down or wasting resources
  - Isolation testing - used to try to zero in on a specific problem in hopes of finding its cause and fixing it
  - Comparative testing - comparing the performance of two or more systems, both to find anomalies and sometimes to make a competitive decision
- Companies such as SmartBear provide applications to perform these kinds of tests

Conclusion

Conclusion

- Embedded Systems development is heavily cost-driven and there are significant market pressures to “shoot the Engineer and put it into production.” However, releasing products that have Safety or Security vulnerabilities can cost hundreds of thousands of dollars in rework or MILLIONS in lawsuits
- Improve Software Quality by following the “10 Steps”
- Slow down and take the time to design and code properly, since *“A majority of security vulnerabilities are a result of coding errors that go undetected in the development stage”* [PR0]
- Employ coding standards such as CERT C and CWE. Programming Research, Critical Software, or Jama Software offer tools to assist with analysis of code to these standards
- The **Engility** graphic shows that DEFECTS act like mines in a minefield. While “Code and Test” methodologies just CLEAR A PATH through the minefield, System overload, operator error, or race conditions could force the system into unexplored territory
- Design rigor can actually save cost in the long run

Help Needed

I am trying to get my Senior Member status.

If you are a Senior Member or Fellow, please see me after the meeting.

Speaker/Author Details

Dwight Bues is a Georgia Tech Computer Engineer with 30+ years' experience in computer hardware, software, and systems and interface design. He has worked in Power Generation, Communications, RF, Command/Control, and Test Systems. He is the author of numerous technical articles in ***Design News****,* ***EE Times****, and* ***IoT Institute*** *magazines and blogs. He has previously presented at* ***DesignWest 2013****,* ***EE Live! 2014****, and* ***ESC Silicon Valley 2016*** *conferences.*

Dwight is a Certified Scrum Master and teaches courses in Architecture, Requirements, and IVV&T. He is also a certified Boating Safety instructor with the Commonwealth of Virginia and the United States Power Squadrons. He is currently working several STEM projects, sponsoring teams for competitions in the Aerospace Industries Association's (AIA) Team America Rocketry Challenge (TARC) and the Robotics Education and Competition Foundation's, Vex Skyrise Robotics Challenge.

Engility Bio

Engility Holdings, Inc. (NYSE: EGL), a leading provider of mission-critical and highly technical services to the U.S. government, is engineered to make a difference. Built on a five-decade commitment to our customers and our country, Engility delivers world-class performance, efficiency and value in a broad range of services, including engineering and technology life cycle support, program and business support and specialized technical consulting. Headquartered in Chantilly, Virginia, and with offices around the world, Engility supports customers throughout the defense, intelligence, space, federal civilian and international communities, drawing on our intimate understanding of customer needs, our deep domain expertise and our highly skilled employees to develop and deliver on-target solutions.

Publication References

Stevens, Richard. Systems Engineering: Coping with Complexity, Prentice Hall, 1998, ISBN 0-13-095085-8

Juran, Joseph and Godfrey, A. Blanton. Juran's Quality Handbook, McGraw Hill, 1998, ISBN 0-07-034003-X

Military and Aerospace Electronics, "Safety and Security –critical Avionics Software," Feb 2011 [MAE]

High Assurance Systems white paper, "Cybersecurity for Things, Part 1," 2017 [HAS]

Chabroux, Michael, "There is no Safety without Security and no Security Without Safety," **Wind River**, 2016 [Wind]

SmartBear, "API Load Testing" [API]

Thomas, Jay, **Embedded Magazine**, "Software Standards 101: Tracing Code to Requirements," 22 Mar 2016 [JT]

Programming Research, "How IoT is Making Security Imperative for all Embedded Software," 2016 [PR0]

Publication References (cont.)

- Beaulieu, Ray (et al), “Simon and Speck: Block Ciphers for the Internet of Things,” 9 Jul 2015, http://csrc.nist.gov/groups/ST/lwc-workshop2015/papers/session1-shors-paper.pdf [S&S]
- **QRA Corp**, “How to Eliminate Over Half of all Design Errors Before they Occur” [QRA]
- **Jama Software**, “Traceability Best Practices” [Jama]
- **Programming Research**, “Addressing Security Vulnerabilities in Embedded, Applications…,” [PR1] and “Automated Coding Standards” [PR2]
- **Critical Software**, “Automated Verification and Validation,” and “Safety Critical Validation” [SCV]
- **Veracode**, “State of Software Security,” 2016 [SoSS]
- **The Atlantic**, The Coming Software Apocalypse”, Sept 2017 [CSA]

Contact Us

Our website:

www.engilitycorp.com

Contact Information:

Dwight Bues
Systems Engineer
(703) 653-5843
Dwight.Bues@engility.com

ENGILITY

Engineered to Make a Difference