LF live

MENTORSHIP SERIES

LF Live MENTORSHIP SERIES

Software Engineering and Functional Safety

Pete Brink, Functional Safety Engineering Leader, kVA by UL

Agenda

- Introduction
- What is Software Engineering?
- The SWEBOK
- How do you know when you’re done?

Safety

- Differences between Quality and Safety
- Safety Analysis
- Code Coverage

- In this presentation we seek to answer the questions:
  - How do you know when you’re done?
  - What is quality and how can we demonstrate it?
  - What is the difference between quality and safety?

Introduction

- Who is Pete?
  - Current: FS Engineering Leader for kVA by UL
  - Past:
    - Jet Engine Control Systems
    - Print Servers
    - Cable Scanners
    - MPLAB
    - Intel – SW Engineering Curriculum, Autonomous Driving
    - Industrial Lasers

- Who is Pete?
  - Software Engineering Program Evaluator for ABET
  - Past:
    - SWECOM Update
    - SWEBOK Update
  - Currently collaborating with the LF and others to create an introductory class (2-3 hours) on SW Engineering
    - Volunteers welcome

- References
  - SWEBOK - SWEBOK Main Page
  - SWECOM - SWECOM Download
  - ABET - ABET Home Page
  - CC2020 - ACM CC 2020

What is Software Engineering?

- What is Software Engineering?
  - According the CC 2020 (from the ACM):
    - “Software engineering (SE) is an engineering discipline that focuses on the development and use of **rigorous methods** for **designing** and **constructing** software artifacts that will **reliably perform** specified tasks.” (page 28)

- Software Engineer vs. Software Engineering
  - CC 2020:
    - “The term “software engineer” (used to denote a profession) is much more broadly employed than “software engineering” as an academic discipline or a degree program. There are many more individuals with a job title or professional identity of a “software engineer” than those who have graduated from software engineering programs.” (page 28)

- What is Software Engineering?
  - Restated by Pete:
    - "Software engineering is the method by which we, the software engineers, control the amount of systematic error we build into the system."

    or

    - "The method by which we can definitively demonstrate that a given software system is complete prior to release."

The SWEBOK and the SWECOM

- SWEBOK
  - The Software Engineering Body of Knowledge
  - Created and maintained by the IEEE-CS
  - Contained in a Wiki Page to be made generally available for anyone to use
  - The SWEBOK is a basic reference outlining all the topics in software engineering

- SWEBOK
  - 15 Categories of skills covering everything from basics of software engineering to software economics

- SWECOM
  - The Software Engineering Competency Model
  - Created and maintained by the IEEE-CS
  - Free to download as a PDF
  - The SWECOM is a competency framework for evaluating skill

- SWECOM
  - 5 competency levels across 13 Categories of skills
  - Skills are grouped into two different categories:
    - Lifecycle Skills
    - Cross Cutting Skills

- Lifecycle Skills:
  - **Requirements**: Elicitation, Analysis, Verification
  - **Design**: Architecture, Design, Patterns, Verification
  - **Construction**: Coding, Debugging, Review
  - **Testing**: Unit, Integration, CI, Defect Tracking
  - **Sustainment**: Support, Maintenance

LF Live MENTORSHIP SERIES

- Cross Cutting Skills:
  - **Process**: SDLC, Assessment and Improvement
  - **Systems Engineering**: Systems Process, Concept
  - **Quality, Safety, Security**: QM, Audit & Assessment
  - **Configuration**: Managing updates and releases
  - **Measurement**: Performance Analysis, Optimization
  - **Human-Computer Interaction**: Usability, Testability

- Competency Model
  - The SWECOM also includes a competency model for each of these skills
    - Technician : Follows
    - Entry Level Practitioner : Assists
    - Practitioner : Participates
    - Technical Leader : Leads
    - Senior Software Engineer : Creates

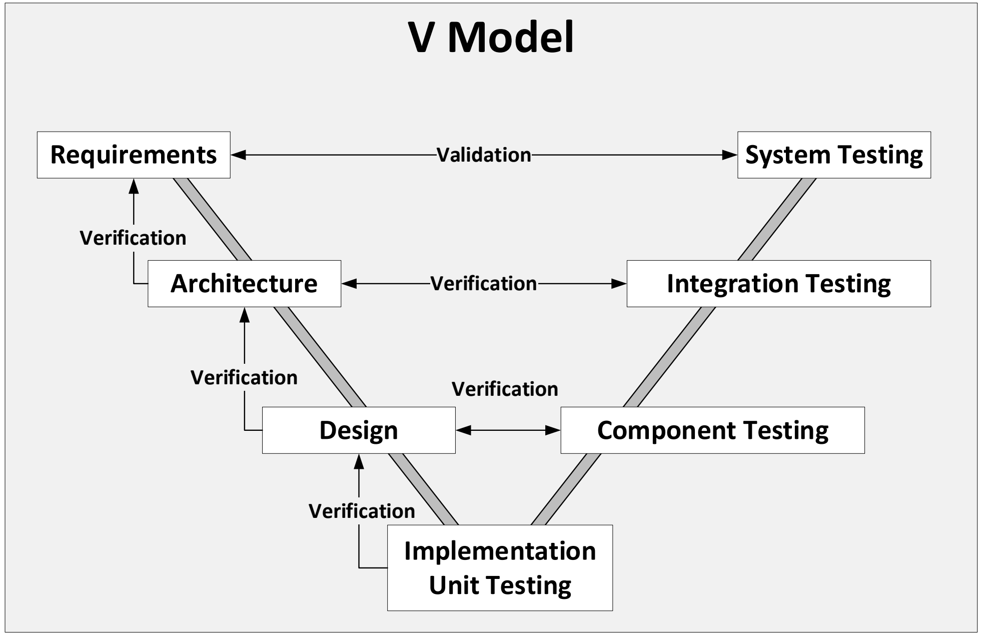
LF Live MENTORSHIP SERIES

Issue: How do you know when you’re done?

- Software Development Life Cycle (SDLC):
  - SDLC defines a framework where there are specific methods for each stage of development
  - "Waterfall", Spiral, Iterative, "Agile"
  - Why do we follow a process?

- Development models exist to demonstrate:
  - Project completion
  - Achievement of stated properties in the final product
  - System properties can be:
    - Quality
    - Safety
    - Security

- System "V" model
  - From Systems Engineering
  - "Outside-In"
  - Can be applied at the system or component level



So…

– How do you know when you’re done?

So…

- How do you know when you’re done?

- You can demonstrate that all the requirements have been fulfilled

Now that we can demonstrate that we are complete, how do we know that the product was built with sufficient quality?

- What is Quality?

- Quality is…
  - Meeting or exceeding your customers expectations (requirements)
  - Bug Rate (a bad metric for controlling quality)
  - Adherence to process (controlling the quality at each step during the lifecycle)

- Why is bug rate a bad metric?

- Why is bug rate a bad metric?
  - **It’s too late**
  - Because software is systemic, we need to get rid of the bugs before we write the code

Safety

- According to ISO 26262:2018, safety-critical software is:
  - SW that enables safe execution of a nominal function
  - SW that enables the system to achieve or maintain a safe state
  - SW that detects, indicates or mitigates HW faults
  - ...

- Most importantly, safety standards expect that all safety software is developed according to a quality standard:
  - ISO 9001 (+IATF 16949)
  - ASPICE (Automotive Software Process Improvement Capability dEtermination)
  - ISO 12207

- Differences between quality and safety:
  - Safety-critical software is expected to go through a safety analysis phase
    - SW FMEA (Failure Modes and Effects Analysis)
    - DFA (Dependent Failure Analysis or Freedom From Interference)

- Differences between quality and safety:
  - Unit verification that cover as close to 100% of the source code as possible (type of coverage varies based on criticality):
    - Statement coverage
    - Branch coverage
    - MC/DC (Modified Condition/Decision Coverage)
  - Call and Function coverage at the architecture level

LF Live MENTORSHIP SERIES

Summary

- Software Engineering is a broad discipline
- Following a systematic approach, we can create software that can demonstrate:
  - Quality
  - Safety
  - Security : follows the same model as safety

LF live MENTORSHIP SERIES

Thank You

Please direct any queries to: peter.brink@ul.com

LF live MENTORSHIP SERIES

Thank you for joining us today!

We hope it will be helpful in your journey to learning more about effective and productive participation in open source projects. We will leave you with a few additional resources for your continued learning:

- The LF Mentoring Program is designed to help new developers with necessary skills and resources to experiment, learn and contribute effectively to open source communities.
- Outreachy remote internships program supports diversity in open source and free software
- Linux Foundation Training offers a wide range of free courses, webinars, tutorials and publications to help you explore the open source technology landscape.
- Linux Foundation Events also provide educational content across a range of skill levels and topics, as well as the chance to meet others in the community, to collaborate, exchange ideas, expand job opportunities and more. You can find all events at events.linuxfoundation.org.