Software Safety

Process Overview and Application

The objective of this course is to provide the learner a comprehensive introduction into *software* safety and how the safety principles in software are applied in a product development environment.

Author/Instructor: Dr. Michael F. Siok, PE, ESEP

Lockheed Martin Aeronautics Company

Lockheed Martin Aeronautics video . . .

Safety and Software

- Lower software defect rates ≠ Safe Software
- Reliable Software ≠ Safe Software
- Secure Software ≠ Safe Software

- What is Safe Software, Software Safety ? ? ? ? ?

- SYSTEMS are safe or not safe
  - *Software enables us to build bigger and/or more complex systems*
  - *Software contributes to System Safety*

Safety and Software

Some misconceptions about safety in software . . .

Lower defect rates . . . While this may improve quality and reliability of the software, improvements in safety of software include the mitigation of anticipated hazards which software may be a part. The hazard may occur because of a specific defect or it may occur because of the current software design and implementation allows the hazard to be raised.

Reliable software . . . Reduction of functional defects to improve reliability may still leave open the mitigation of hazards which software plays a part. The software may perform a specific function reliably every time but if this function allows a hazard to be raised, the hazard will be raised reliably as well.

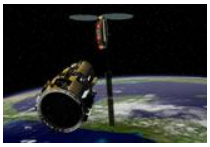
Secure software . . . Reduction of vulnerabilities and certain functional defects to improve security may still leave open the mitigation of hazards which software plays a part. The software may perform a specific function securely every time but if this function allows a hazard to be raised, the hazard will be raised in spite of the security measures in place.

Software is a sequence of machine instructions only. As such, software by itself is not safe or unsafe. Operational SYSTEMS are safe or unsafe due to the nature of their processing and interaction with their environment. It's the operational states of the machine and the environmental interaction which together determine the safety of the system and then, the software.

Software Failures Affect Society

… a few examples

- **"A software glitch, subsequent navigation errors, and excessive fuel use led to failure of an automated NASA spacecraft designed to rendezvous with a Pentagon satellite without human help last year . . ."**



- **"Software Failure Causes Airport Evacuation . . .**
  - ***. . . Normally the software flashes the words "This is a test" on the screen after a brief delay, but this time the software failed to indicate that . . . ."***
- **"Software failure cited in August northeast US electrical system blackout investigation**
  - ***. . . A malfunctioning alarm system controlled by software may have played a big role in the outage . . . ."***

Software Failures Affect Society

1. Software error, subsequent navigation errors, and excessive fuel use caused loss of spacecraft, in April 2005 -- $110M mission. Velocity computations in GPS receiver (.6 meters per second) persisted through aircraft resets because needed patch wasn't installed

2. Atlanta airport evacuation April 21, 2006 due to false alarm in baggage screen system computer; baggage handler stopped the conveyer and looked for the suspicious device for 2 hours shutting down airport, loss of revenue in the commercial airline system, time/delays/rescheduling, etc.

3. August 14, 2003, blackout crippled most of the US NE and parts of Canada causing loss of revenue (many companies), summer heat wave compounding issue, . . . what about hospital operation? Alarm and event processing software routine malfunction did not alert operators to problem which then cascaded through system due to lack of preventative action. Some additional human factors adjustments were discovered needing attention later . . . .

Software Failures Affect Society

… a few examples

- **Air traffic controllers lost voice contact with 400 aircraft over Southwestern U.S. when the *Voice Switching Control System failed* because a 32-bit countdown timer reached zero . . .**
- **Hatch nuclear power plant was forced into emergency shutdown for 48 hours due to a software update to a business network computer . . .**

- **One line of code error in AT&T telephone switch caused cascading failure of telephone switches shutting down AT&T telephone network for 9 hours . . . .**

Software Failures Affect Society

1. 3+ hour shut down of southern California airports. Alleged 5 planes lost required separation distance. 450 flights at LAX diverted; about 30,000 people affected. LAX, Ontario, Bob Hope, San Diego airports experienced delays. The back-up system failed too.

2. March 7, 2008, unit #2 at Hatch. Software update to a business computer but that computer was also used to monitor chemical and diagnostic data from the facility's primary control system. The update was designed to synchronize data on both systems. Rebooting computer data to be reset on the control system causing safety systems to interpret the data as a drop in water reservoirs used to cool the reactors. Triggered shutdown.

3. January 15, 1990, 2:25 in the afternoon, phone switch failure runaway occurred. About 9 hours later systems started to stabilize (night time) – AT&T estimated losing about $60M in unconnected calls. We don't really know the secondary costs to business affected by the outage.

Software Failures Affect All of Us

- These system failures were not planned by their development teams
  - *. . . but they were 'built-in'*
- As we look at what might lie ahead, how can the software industry provide assurance that these types of issues are avoided?

Automated "Defense" System

Self-Driving Cars

Robotic Arthroscopic Surgery

Software Failures Affect All of Us

It should be noted that the software developers did not intentionally build the system to exhibit this abnormal behavior under these special conditions noted. However, the software practice in place allowed the developers to design in these failures . . . unintentionally.

Just a little food for thought . . . Consider some of the systems we will be building in the not too distant future and possible malfunction behavior . . .

1. Automated defense system
2. Self-driving cars (and the safety of your family in that car or another car on the same road)
3. Robotic surgery – pictured robotic arthroscopic surgery.

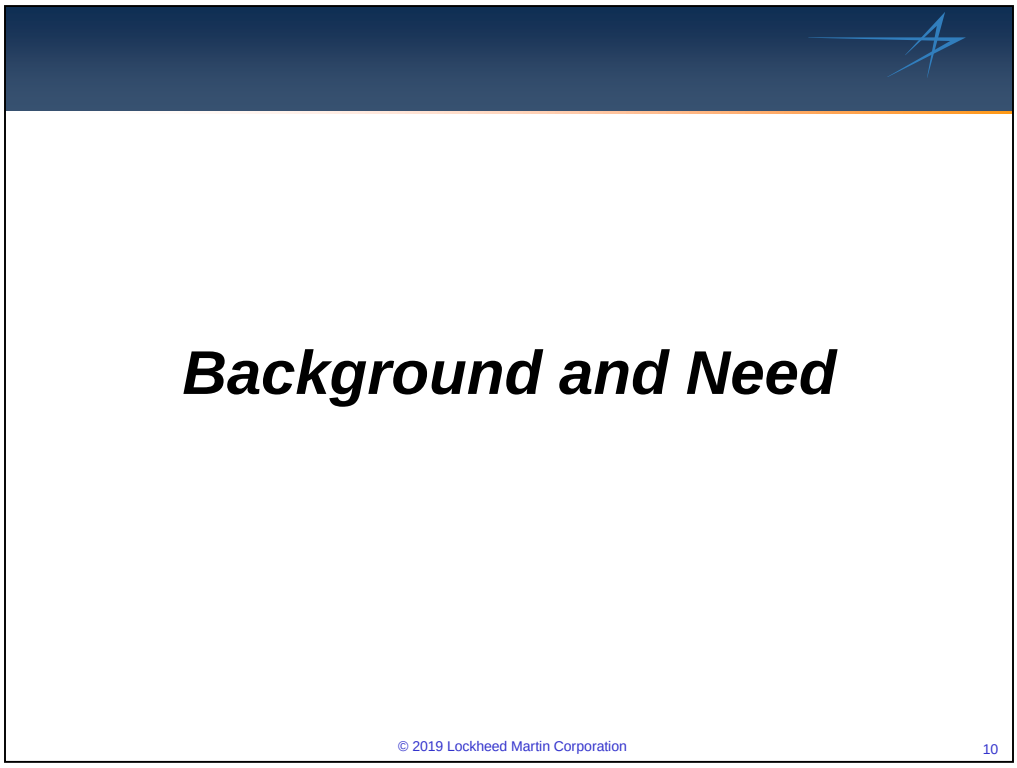
Course Objectives

Introduce the students to materials and guidance available to practicing engineers.

Introduce the students to a software safety practice.

Introduce students to the need for tailoring the software safety process and discuss the general tailoring approach.

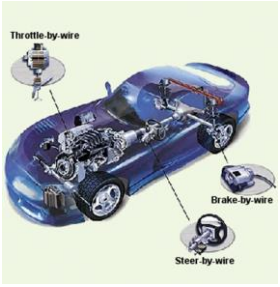
Reinforce learning objectives with a small project example.



Background and Need

Background and Need

- **Software Safety can only be considered in context of an *Operational System***
  - ***Auto/aircraft anti-lock brakes***
  - ***Vehicle Escape System***
  - ***Fly/drive by wire System***
  - ***Traffic Light***
  - ***Heart pacemaker***
  - ***Insulin pump***
  - ***Many, many others . . . .***



- **All have critical software processing that . . . *commands, controls, and/or monitors critical functions necessary for continued safe operation of that system***

Background and Need

Software safety is an attribute of a system and can only be considered within the context of the operational system. Example systems are provided.

Key definition . . .

Background and Need (Cont'd)

- **Definitions:**
  - ***Safety-Critical Software***
    - A software unit, component, object, or software system whose ***proper recognition, control, performance, or fault tolerance is essential to the safe operation*** and support of the system in which it executes.
  - ***Safety-Critical Functions***
    - Any function or integrated functions implemented in software that ***contributes to, commands, controls, or monitors system level safety-critical functions needed to safely operate*** or support the system in which it executes.

Background and Need

Key definitions . . .

1. Safety-Critical Software
2. Safety Critical functions

Background and Need (Cont'd)

- **LM Aircraft systems already have requirements of safety**
  - ***F-16***
  - ***F-22***
  - ***C130***
  - ***C-5***
  - ***F-35***
  - ***UAV***

- **Customer requirements for safety usually specified in contracts**
  - ***E.g., MIL-STD 882, ARP-4761***
  - ***Software not excluded from safe systems operation***

Mil-STD 882: Department of Defense Standard Practice for System Safety
Aerospace Recommended Practice ARP-4761: Guidelines and Methods for Conducting the Safety Assessment

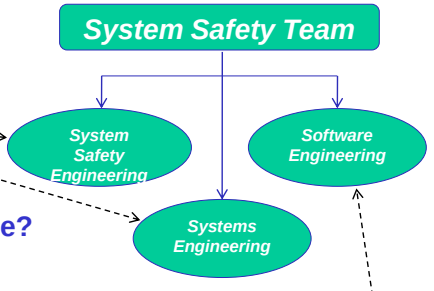
13

Background and Need

LM Aeronautics Company aircraft programs have requirements for safety and safety in software. Customer contracts usually require MIL STD 882 and/or ARP-4761. These standards provide guidance to contractors in defining and implementing a system safety program encompassing the product . . . including software.

How Do We ID Critical Software Processing?

- **DEFINITION: Software Safety -- application of disciplined safety engineering, systems engineering, and software engineering practices to be sure that active measures are taken to assure system integrity through prevention, elimination, and/or control of hazards that may be caused or induced by . . . *Software*.**



- **How to ID critical processing?**
  - ***Hazard Analysis***
- **How to Provide SW Safety Assurance?**
  - ***SW Architecture & Design***
  - ***SW Processes & Methods***
  - ***SW Tooling***

How Do We Identify Critical Software Processing?

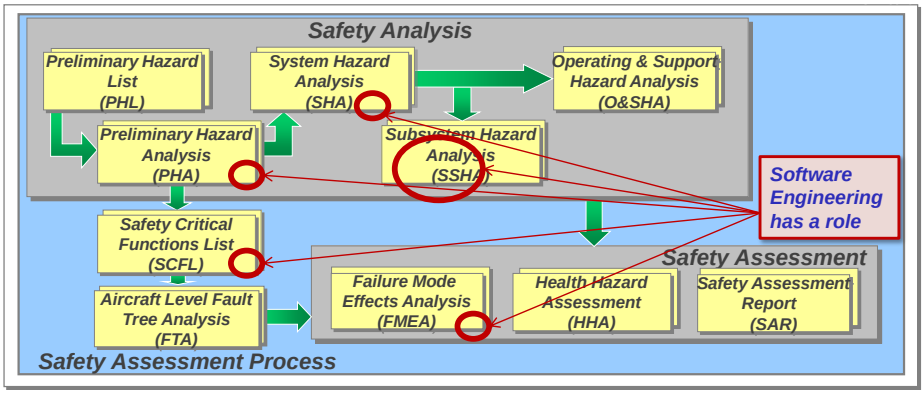
Organization

Process

Methods

Hazard Analysis

- **System Safety analysis method to . . .**
  - *Identify hazards to system, mission, or element*
  - *Assess severity, likelihood of occurrence, & consequences of each hazard on affected system elements*
  - *Identify safety requirements & preferred designs.*



Hazard Analysis

What is it?

Background and Need (Cont'd)

- **Goal of Software System Safety Program**
  - ***Integrate seamlessly with System Safety Program***
  - ***Reduce risk of serious hazards caused by/induced by software to acceptable levels***
    - As Low As Reasonably Practicable (ALARP)
      - *Judgment of balance of risk and societal benefit*
      - *Risk must be insignificant in relation to time, money, and effort to avert it*
      - *Is "good engineering practice" enough?*
- **System Safety Program**
  - ***Identifies possible hazards to aircraft, mission, and/or environment***
  - ***Assesses severity, likelihood of hazard occurrence, and likely consequences***
  - ***Assesses and implements actions to manage risk***
  - ***Specifies safety requirements***
  - ***Reviews preferred design approaches***
  - ***Reviews discovered faults and failures affecting safety critical systems (and software) and their repair action status***
  - ***Assesses safe flight readiness***

Background and Need

Software safety must operate in tandem with system safety program.

Background and Need (Cont'd)

-- MIL-STD 882E Mishap Severity Categories

- **MIL-STD 882E, 5/11/2012**
  - *Systems engineering approach to eliminate system hazards and minimize risks where hazards cannot be eliminated*
  - *Version 'E' includes handling of software*
  - *Quick review . . . Hazards are assigned severity . . .*

Severity Categories		
Description	**Severity Category**	**Mishap Result Criteria**
Catastrophic	1	Could result in one or more of: death, permanent total disability, irreversible significant environmental impact, or monetary loss equal to or exceeding $10M
Critical	2	Could result in one or more of: permanent partial disability, injuries or occupational illness affecting at least 3 people, reversible significant environmental impact, or monetary loss $1M ≤ *x* <$10M
Marginal	3	Could result in one or more of: injury or occupational illness resulting in loss of 1 or more work days, mitigatable moderate environmental impact, or monetary loss $100K ≤ *x* <$1M
Negligible	4	Could result in one or more of: injury or occupational illness not resulting in lost workdays, minimal environmental impact, or monetary loss less than $100K

17

Background and Need

MIL-STD 882E mishap severity levels definitions – right out of the standard.

Background and Need (Cont'd)

-- MIL-STD 882E Probability Levels

- **How often we expect the hazard to occur . . .**

Probability Levels				
Description	**Level**	**Specific Item**	**Fleet [1]**	**Probability of Occurrence [2]**
Frequent	A	Likely to occur often in the life of the item.	Continuously experienced.	$x \geq 10^{-1}$
Probable	B	Will occur several times in the life of the item.	Will occur frequently.	$10^{-1} > x \geq 10^{-2}$
Occasional	C	Likely to occur sometime in the life of the item.	Will occur several times.	$10^{-2} > x \geq 10^{-3}$
Remote	D	Unlikely, but possible to occur in the life of the item.	Unlikely, but can reasonably be expected to occur.	$10^{-3} > x \geq 10^{-6}$
Improbable	E	So unlikely, it can be assumed occurrence may not be experienced in the life of the item.	Unlikely to occur, but possible.	$x < 10^{-6}$
Eliminated	F	Incapable of occurrence. This level is used when potential hazards are identified and later eliminated.		

NOTES:
1 - Fleet size should be defined
2 - Probability of Occurrence = (number of events) / (specific exposure (e.g., number of A/C, FH, Years of service, etc.))

Background and Need

Probability levels right out the standard. The probability of occurrence column was taken from Appendix A and blended with the same table in the standard.

Background and Need (Cont'd)

-- MIL-STD 882E Risk Assessment

- **Hazard Risks are identified by Risk Assessment Code (RAC)**
  - ***Combination of severity category and probability of occurrence***

Risk Assessment Matrix				
Severity / Probability	Catastrophic (1)	Critical (2)	Marginal (3)	Negligible (4)
Frequent (A)	High	High	Serious	Medium
Probable (B)	High	High	Serious	Medium
Occassional (C)	High	Serious	Medium	Low
Remote (D)	Serious	Medium	Medium	Low
Improbable (E)	Medium	Medium	Medium	Low
Eliminated (F)	Eliminated			

- **However, software risk assessments cannot rely solely on severity and probability**
  - **Reliability of SW not estimated like HW Reliability**
  - **Assess SW contribution to system risk using severity and SW 'degree of (automated) control' – (Software Control Categories) --**

19

Background and Need

Hazard risk matrix from the standard. This is mapped differently than the previous 882D but essentially does the same job – categorizes the severity and probability of a hazard into a risk matrix and color codes the risk assessments.

Background and Need (Cont'd)

-- MIL-STD 882E Software Control Categories

Software Control Categories		
Level	**Name**	**Description**
1	Autonomous (AT)	SW functionality that exercises autonomous control authority over potentially safety-significant HW systems, subsystems, or components without possibility of predetermined safe detection and intervention by a control entity to preclude occurrence of the mishap or hazard.
2	Semi-Autonomous (SAT)	1. SW functionality that exercises control authority over potentially safety-significant HW systems, subsystems, or components allowing time for predetermined safe detection and intervention by independent safety mechanisms to mitigate or control the mishap or hazard. 2. SW item that displays safety-significant information requiring immediate operator entity to execute predetermined action for mitigation or control over the mishap or hazard. SW exception, failure, fault, or delay will allow, or fail to prevent, mishap occurrence.
3	Redundant Fault Tolerant (RFT)	1. SW functionality that issues commands over safety-significant HW systems, subsystems, or components requiring a control entity to complete the command function. The system detection and functional reaction includes redundant, independent fault tolerant mechanisms for each defined hazardous condition. 2. SW that generates information of a safety-critical nature used to make critical decisions. The system includes several redundant, independent fault tolerant mechanisms for each hazardous condition, detection, and display.
4	Influential	SW generates information of a safety-related nature used to make decisions by the operator, but does not require operator action to avoid a mishap.
5	No Safety Impact (NSI)	SW functionality that does not possess command or control authority over safety-significant HW systems, subsystems, or components and does not provide safety-significant information. SW does not provide safety-significant or time-sensitive data or information that requires control entity interaction. SW does not transport or resolve communication of a safety-significant or time sensitive nature.

Background and Need

For 882E, the Software Control Categories (SCC) are new. The table above presents the SCCs and the definition of each category as stated in the standard. The definitions are kind of dry and generic as presented here.

Background and Need (Cont'd)

-- MIL-STD 882E Software Control Categories

Software Control Categories		
Level	**Name**	**Considerations**
1	Autonomous (AT)	● Failure directly results in a mishap ● No possibility of operator action to prevent the mishap.
2	Semi-Autonomous (SAT)	● Failure could directly result in mishap if operator does not act ● There is time for predetermined safe detection and intervention by independent safety mechanisms to mitigate or control the mishap
3	Redundant Fault Tolerant (RFT)	● System detection and functional reaction includes redundant, independent fault tolerant mechanisms for each defined hazardous condition ● SW with a failure condition requires another independent fault to result in a mishap
4	Influential	● SW with a failure condition that reduces redundancy or safety margins but at least one independent mechanism remains to preclude a mishap ● Operator makes the decisions
5	No Safety Impact (NSI)	● After a SW failure there still are at least two independent mechanisms to preclude a mishap

- **Software Control Categories (SCC) identify degree of software (automated) control involved in hazard**
- **SCC listed in order top to bottom, most software automated control to least**
- **Considerations more simply describe failure, detection, and intervention behavior for SCC level**
- **Software safety criticality characterized by "severity category" and "level of software control"**

Background and Need

Here the SCCs are defined a little differently, hopefully in a way that provides for context around what we do here at LM Aero. The SCCs are listed in order of decreasing level of control from top to bottom.

Background and Need (Cont'd)

-- MIL-STD 882E Software Criticality Index and Level of Rigor

Software Safety Criticality Matrix				
SW Control Category	Severity Category			
	Catastrophic (1)	Critical (2)	Marginal (3)	Negligible (4)
1	SWCI 1	SWCI 1	SWCI 3	SWCI 4
2	SWCI 1	SWCI 2	SWCI 3	SWCI 4
3	SWCI 2	SWCI 3	SWCI 4	SWCI 4
4	SWCI 3	SWCI 4	SWCI 4	SWCI 4
5	SWCI 5	SWCI 5	SWCI 5	SWCI 5

SWCI	Level of Rigor Tasks
SWCI 1	Program shall perform analysis of requirements, architecture, design, and code and conduct in-depth safety-specific testing.
SWCI 2	Program shall perform analysis of requirements, architecture, design, and conduct in-depth safety-specific testing.
SWCI 3	Program shall perform analysis of requirements and architecture and conduct in-depth safety-specific testing.
SWCI 4	Program shall conduct safety-specific testing.
SWCI 5	Once assessed by safety engineering as Not Safety, then no safety specific analysis or verification is required.

- **Software Safety Criticality Matrix (SSCM) maps SCCs to severity categories to identify Software Control Index (SWCI)**
- **SWCI identifies most critical (SWCI 1) to least critical (SWCI 5), not color coded**
- **SWCI maps to Level of Rigor (LoR) tasks**
- **Successful execution of LoR tasks increases confidence software will perform as specified**

Background and Need

Using the SCC and the Mishap Severity Levels, a Software Control Index (SWCI) is assigned to a software risk item. The SWCI maps directly to a Level of Rigor (LoR) assignment. The LoR is the degree to which software processes are applied to reduce the risk of the assessed hazard from occurring due in part or in total to software causes.

Background and Need (Cont'd)

-- MIL-STD 882E SWCI, Risk, LOR, and Consequences

Relationship between SWCI, Risk Level, LOR, and Risk		
SWCI	Risk Level	SW LOR Tasks and Risk Assessment/Acceptance
SWCI 1	High	If SWCI 1 LOR tasks are unspecified or incomplete, the contributions to system risk will be documented as HIGH and provided to the PM for decision. The PM shall document the decision of whether to expend the resources required to implement SWCI 1 LOR tasks or prepare a formal risk assessment for acceptance of a high risk.
SWCI 2	Serious	If SWCI 2 LOR tasks are unspecified or incomplete, the contributions to system risk will be documented as SERIOUS and provided to the PM for decision. The PM shall document the decision of whether to expend the resources required to implement SWCI 2 LOR tasks or prepare a formal risk assessment for acceptance of a SERIOUS risk.
SWCI 3	Medium	If SWCI 3 LOR tasks are unspecified or incomplete, the contributions to system risk will be documented as MEDIUM and provided to the PM for decision. The PM shall document the decision of whether to expend the resources required to implement SWCI 3 LOR tasks or prepare a formal risk assessment for acceptance of a MEDIUM risk.
SWCI 4	Low	If SWCI 4 LOR tasks are unspecified or incomplete, the contributions to system risk will be documented as LOW and provided to the PM for decision. The PM shall document the decision of whether to expend the resources required to implement SWCI 4 LOR tasks or prepare a formal risk assessment for acceptance of a LOW risk.
SWCI 5	Not Safety	No safety-specific analysis or testing is required.

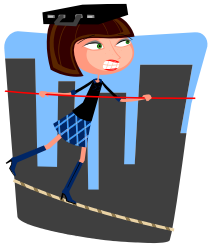
Background and Need

Based on the quality of the LoR implementation by the software team (through review or other assessments) will determine the risk level associated or assessed with the particular hazard with this SWCI. The risk for the hazard is monitored singly based on this assessment for software LoR. It is possible for multiple SWCIs of the same level to be rated differently or the same.

Background and Need (Cont'd)

-- MIL-STD 882D Mishap Severity Categories (Cont'd)

- **3 Assessment Areas for Safety Risk Consequence**
  - ***Person or people***
    - Death
    - Disability
    - Injury, Illness
    - Lost work
  - ***Financial Loss***
    - $ millions or more
    - Negligible
  - ***Damage to Environment***
    - Irreversible or reversible severe damage
    - Break Regulations or Laws
    - Affect protected species, land, water, resources, etc.



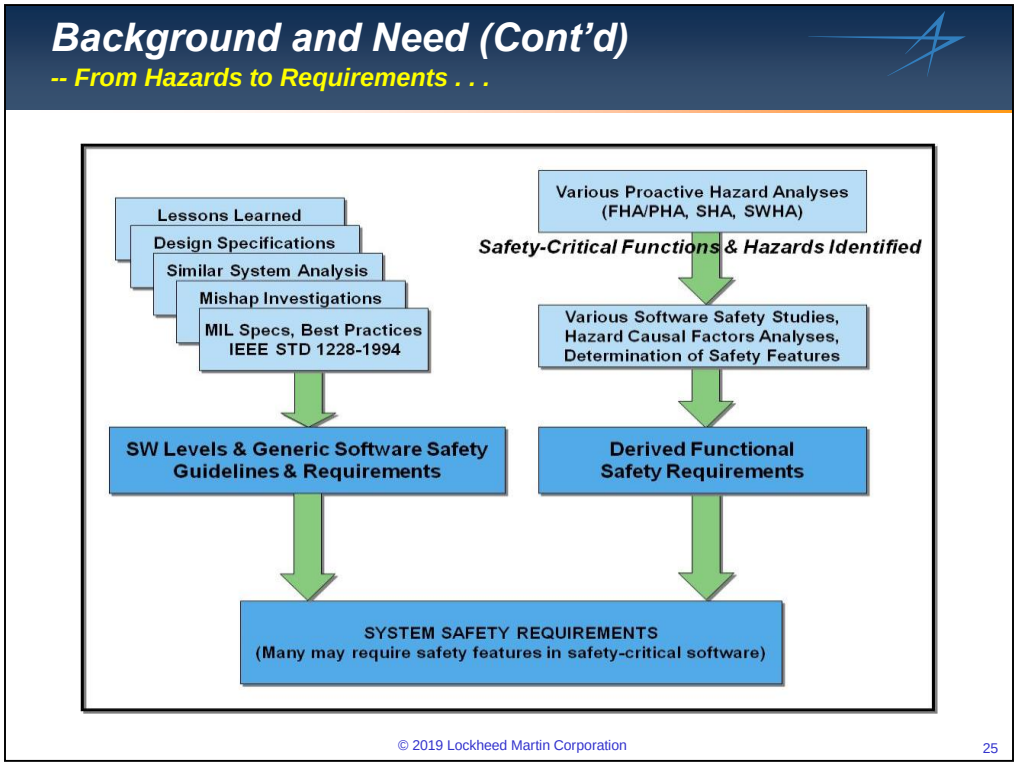
Background and Need

3 areas for safety consequence:

People

Money

Collateral damage to environment.



Background and Need

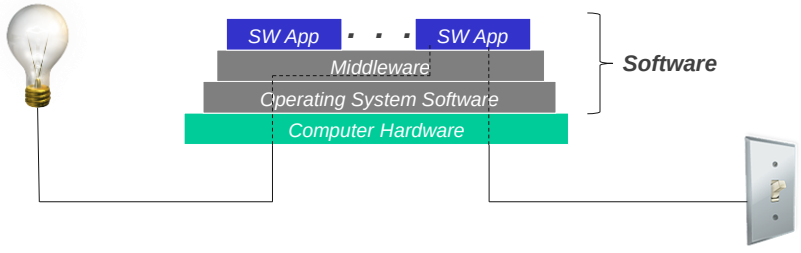
Where do safety requirements come from?

Background and Need (Cont'd)

-- OK . . . So What About Software Safety Now?

"Reason Model" of Organizational Accident Causation (James Reason, 1990, 1991).

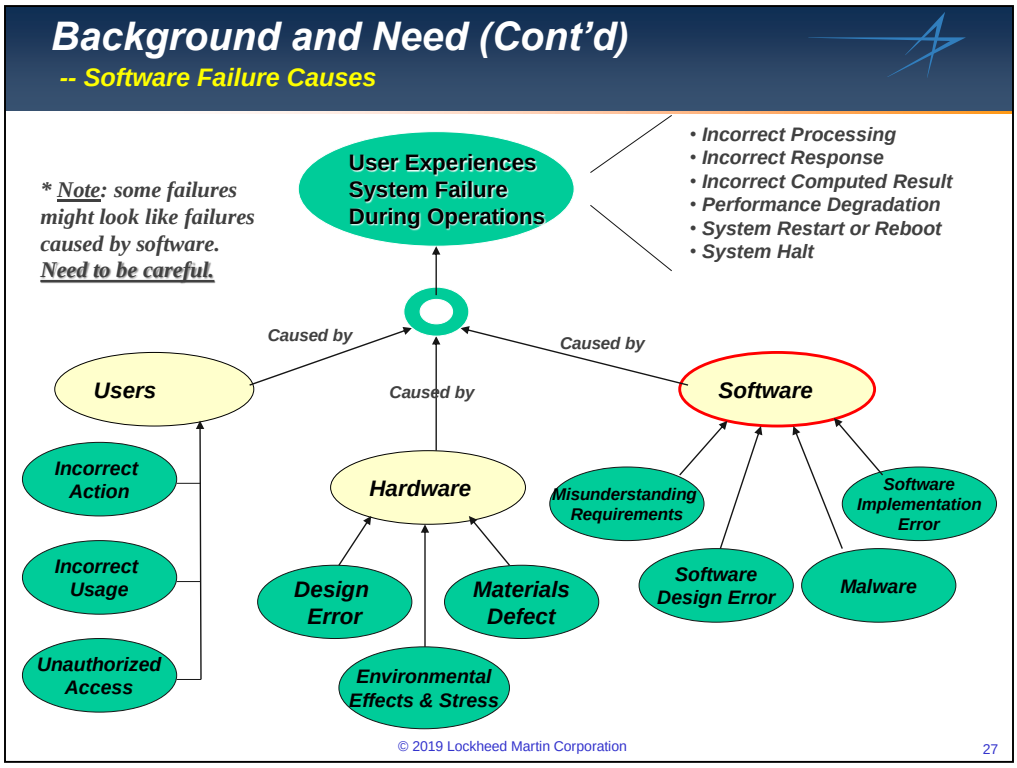
How can *Software* cause mishaps or accidents???



Background and Need

Reason's Model of organizational accident causation shows that there are many part of a design space, some independent. If issues line up just right between all the components, this could lead to a mishap through a chain of events.

Discuss how to tune on the light from a computer controlled light switch.



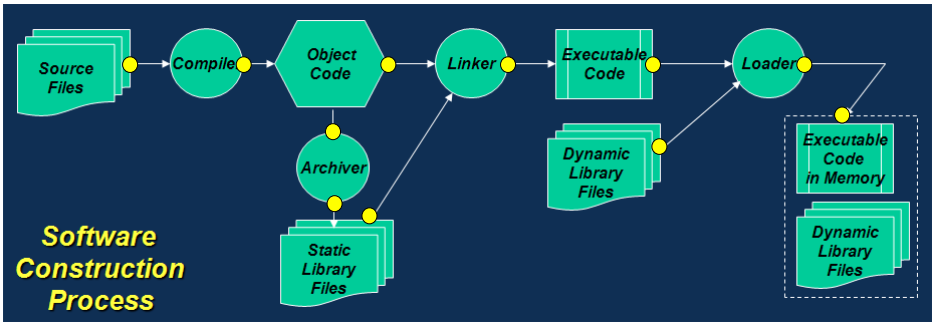
Background and Need

Software failure causes may not be so apparent. Failures can originate in many places. Then what is failure?

Background and Need (Cont'd)

-- Sources of Errors in Software Process

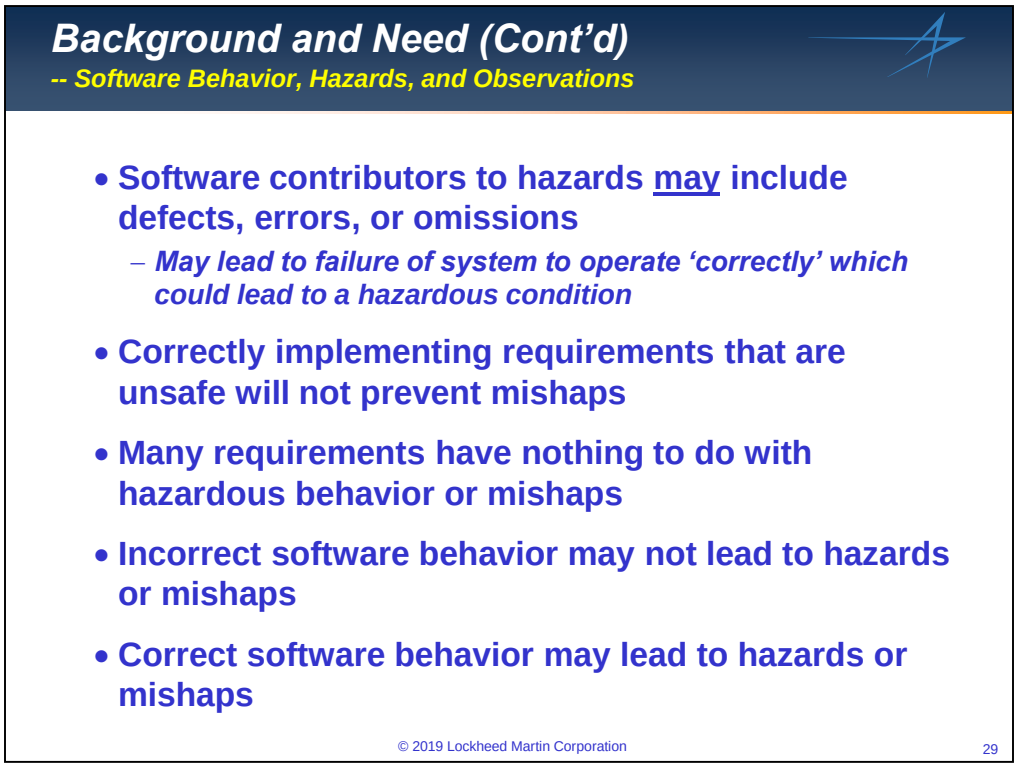
- **Causes of Software failures**
  - *Latent defects in the source code, library files*
  - *Latent defects in tools affecting code construction*
  - *Environmental conditions operational software is not programmed to handle*



Sources of error injection

Background and Need

Sources of errors in software. Review general software construction process.



Background and Need

Software behavior, hazards, and observations:

Just a note here about looking for hazards:

-- Not all software error or faults lead to hazards or mishaps

-- Correctly implementing requirements known or unknown to be unsafe will not prevent mishaps

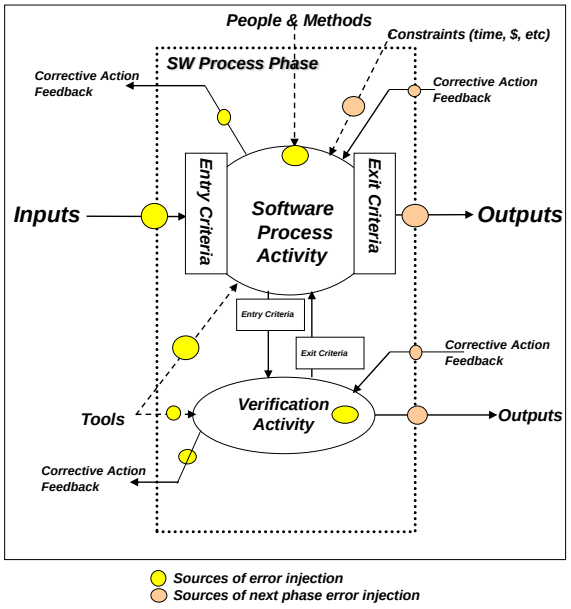
-- Typically hazards rise out of smaller or limited parts of the system. Many of our requirements have nothing to do with hazardous system behavior or lead to mishaps (by themselves)

-- Incorrect software behavior may not lead to hazards or mishaps – these may occur in parts of the system where their incorrect behavior does not present a hazard or threat to the system operational context.

Bottom line – It is hard to find all the hazards of a system.

Background and Need (Cont'd)

-- Sources of Errors in Software Process

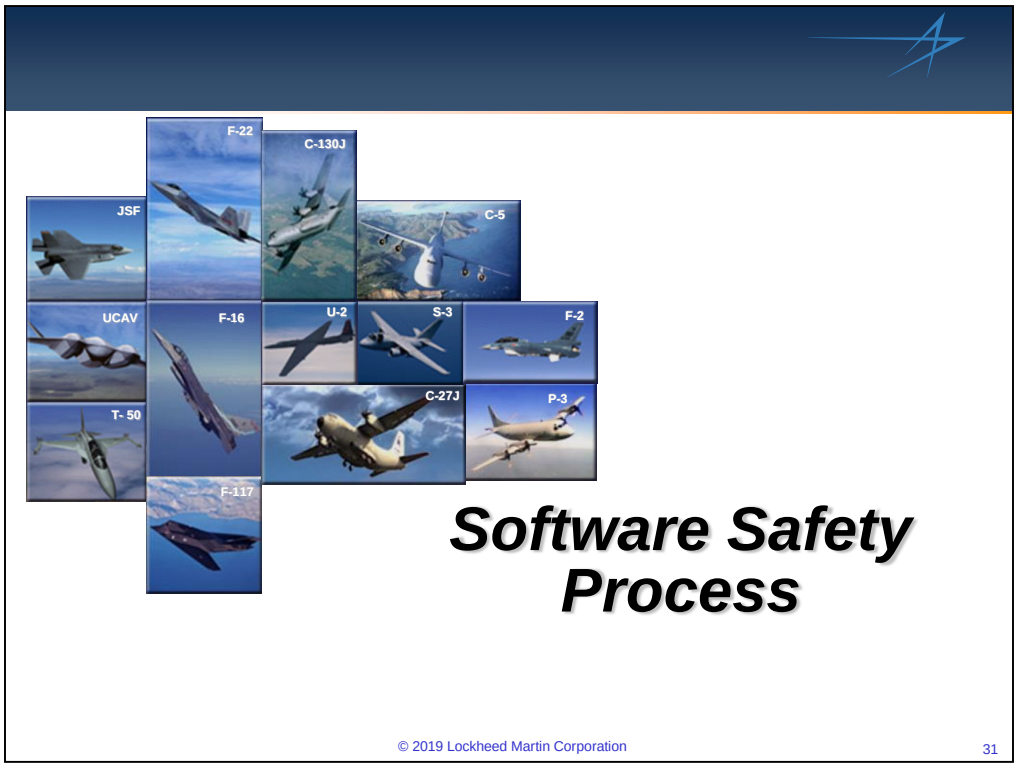


- **Software Safety is not only about reducing error rates in safety-critical software (based on SCC)**
- **Software Safety is also about reducing the risk of software causing or inducing certain hazards that when realized, could lead to a system mishap, accident**

Background and Need

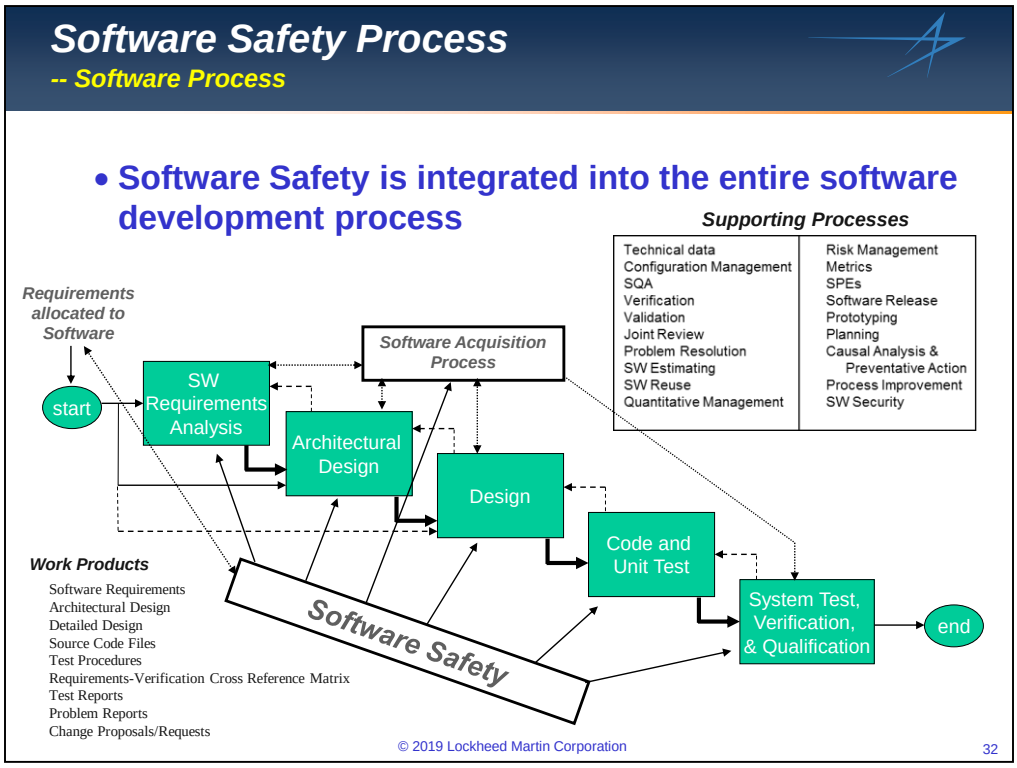
Software safety quick review:

1. Conduct hazard analysis
2. Isolate SC pieces and create assurance level based on criticality.



Software Safety Process

An overview of the software safety process . . . .

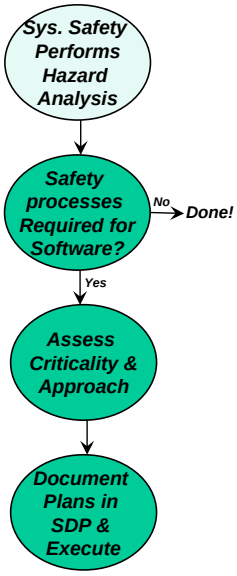


Software Safety Process

Software Safety must be integrated into the software lifecycle, including flow-down to subs, as applicable.

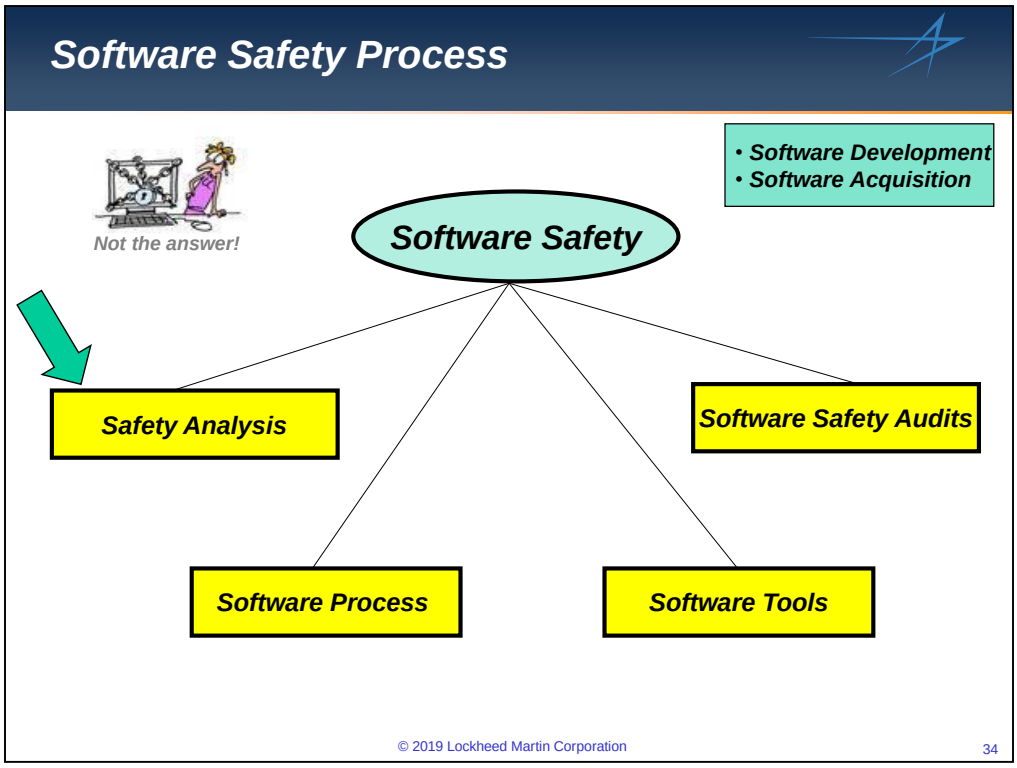
Software Safety Process
-- General Approach

- **General Approach to safety in software . . .**
  - ***Participate in System Safety Analysis Activities***
    - Hazard analysis and other system safety team sponsored analyses
  - ***Adjust software process Level of Rigor (LOR) activities and/or software product design activities based on "Levels of Safety Criticality"***
    - Document in SDP
  - ***Address software tool integrity***
    - Document in SDP
  - ***Provide audit trail (process evidence) validating software process and product development & technical integrity***
    - Document in SDP
- **Software Development Plan (SDP) and/or Software Acquisition Management Plan (SAMP) documents project approach to safety in software**



Software Safety Process

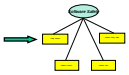
Simple 4-step approach . . . .



Software Safety Process

The discussion that follows is divided into 4 sections plus an overview of development and acquisition.

Software Safety Process
-- Safety Analysis

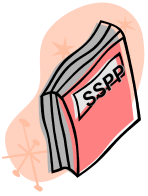


- **Software Engineering organization teams with Safety Engineering to understand hazards (i.e., risks and consequences) due to safety-critical functions failing**
  - ***Some critical functions may be monitored, controlled by software***
  - ***Safety Engineering Team responsible for hazard analysis***
    - Software team offers perspective on likely risks due to software

Hazards List

- **Leads to list of safety-critical functions, level of criticality, and software components that require special handling**

Description	*Criticality*	*Software Component*
Function 1	*Level 1*	*Comp 1, 3*
Function 2	*Level 1*	*Comp 1*
Function n	*Level 3*	*Comp 6, 8*



35

Software Safety Process

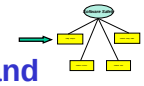
Safety Analysis:

Software engineering helps system safety understand SW architecture and software role in functions

SW engineering consults with system safety on hazard analysis

Output of the safety analysis activity is a list of safety-critical functions, the criticality of these functions, and the list of software components that support the safety critical functions.

Software Safety Process
-- Safety Analysis (Cont'd)



- **SW Engineering helps Safety Team identify appropriate risk reduction techniques to hazards and safety requirements through combination of . . .**
  - ***Software Analysis and Design Choices***
    - Safety-critical software identification
    - Safety interlocks, HW/SW Trades, partitioning, fault tolerance, etc.
    - Requirements, Design, and Coding Standardization
    - Safety Methods for software (SFTA, SFMEA, others)
  - ***Software Process Choices***
    - Defect management
    - Historic and predictive metrics
    - Reuse management, Defect Prevention, Requirements Traceability, etc.
  - ***Tool Choices and Tool Management***
    - Tool configuration control (IDEs, test tools, utilities, etc.)
    - Switch settings, automation choices and maintenance
  - ***Software Product Assurance***
    - Mark specific software and work products
    - Safety Audits
    - Verification, Qualification

Level Of Rigor

SFTA – Software Fault Tree Analysis
SFMEA – Software Failure Modes and Effects Analysis

Software Safety Process

Safety Analysis (cont'd):

These things listed are part of the software engineer's toolbox to help in hazard mitigation and/or risk reduction for safety in software . . . .

The collection of practices and agreements going forward are documented in the **System Safety Program Plan** and the program **Software Development Plan**.

Software Safety
-- Safety Analysis (Cont'd)

- **Safety analysis activities lead to . . .**
  - ***Safety Critical Functions (SCF) List***
  - ***Hazards List***
  - ***Safety Critical Software Components List, with criticality***
  - ***Level of Rigor for SW development tasks***
  - ***System Safety Program Plan (SSPP)***

SWCI	Level of Rigor Tasks
SWCI 1	Program shall perform analysis of requirements, architecture, design, and code and conduct in-depth safety-specific testing.
SWCI 2	Program shall perform analysis of requirements, architecture, design, and conduct in-depth safety-specific testing.
SWCI 3	Program shall perform analysis of requirements and architecture and conduct in-depth safety-specific testing.
SWCI 4	Program shall conduct safety-specific testing.
SWCI 5	Once assessed by safety engineering as Not Safety, then no safety specific analysis or verification is required.

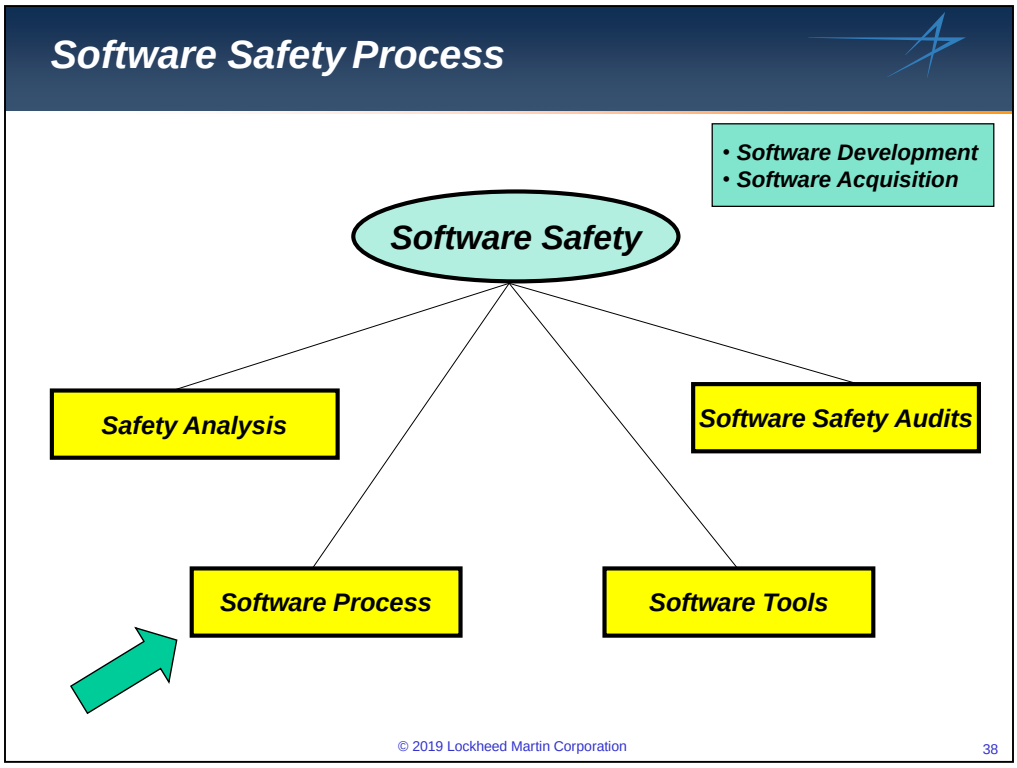
Software Safety

Safety Analysis (cont'd):

From the safety analysis activity, the following are produced:

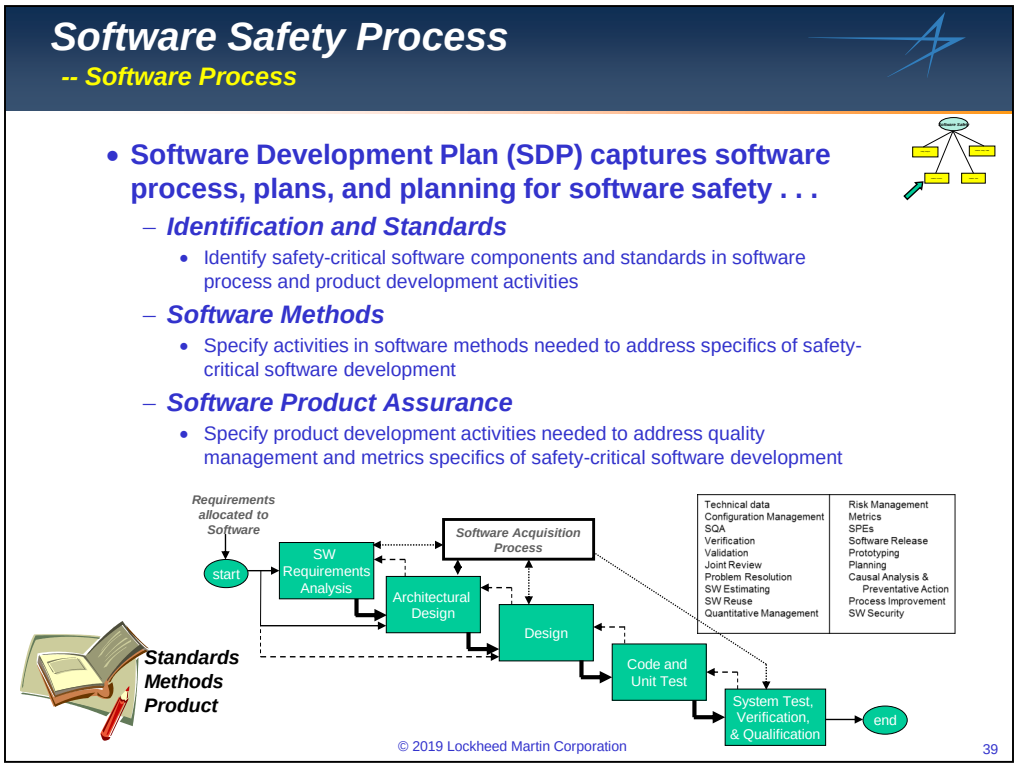
1. Safety Critical Functions Listing
2. Hazard list (initial)
3. Safety critical software components list
4. LoR definitions for the project and defined in the software development plan
5. System Safety Program Plan.

Note: the above data items may exist in various and evolving forms of maturity throughout the program.



Software Safety Process

Software Process



Software Safety Process

Software Process:

Discussion is divided into 3 subsections:

- Identification and standards
- Software Methods
- Software Product Assurance

Software Safety Process
-- Software Process

- **Identification and Standards . . .**
  - *ID Software components to which safety processes apply*
  - *ID Levels of criticality for each identified component*
  - *ID and describe Architectural constraints*
    - Partitioning of software to nodes or address spaces
    - Processing resource allocations and timing
    - Others . . . .
  - *ID Requirements and design standards used for software*
  - *ID Programming languages, coding standards used for software components developed for safety application*
  - *ID Engineer training requirements for development of safety-critical software; schedule training*
  - *ID Role of software safety engineer on software team*
  - *ID Software work products for safety audit*

Standards
Methods
Product

Software Safety Process

Software Process:

Identification & Standards

- Uniquely identify artifacts as safety critical, including level of criticality
- Identify architectural decision, constraints
- Requirements standards
- Design standards
- Coding Standards
- Training
- Software work team includes Software Safety Engineer designate
- Identify Software lifecycle work products available for audit

Software Safety Process
-- Software Process

- **Software Methods**
  - ***Bi-directional Traceability of software safety requirements***
    - Requirements to design to code to test procedures
    - Test procedures to . . . requirements
  - ***Causal Analysis and Preventative Action activities***
  - ***Decision management process for reuse, use, and readiness of safety-critical software***
  - ***Joint review of software products involving application of safety***
    - Reviews with System Safety, Systems Engineering . . . . as applicable
  - ***Prototype software components built in support of safety-critical software development to same LOR***
  - ***Test schedules and resources for safety-critical software***
  - ***Inspection or walkthrough review methods for each software work product involving safety-critical software***
  - ***Requirements and process for reuse of safety-critical software***
    - Including reuse of requirements, design, and test work products as well as code, distribution, licensing, etc . . .
  - ***Impact analysis on proposed changes to safety-critical software***
    - Perform updates with same process rigor used during initial software development unless documented otherwise

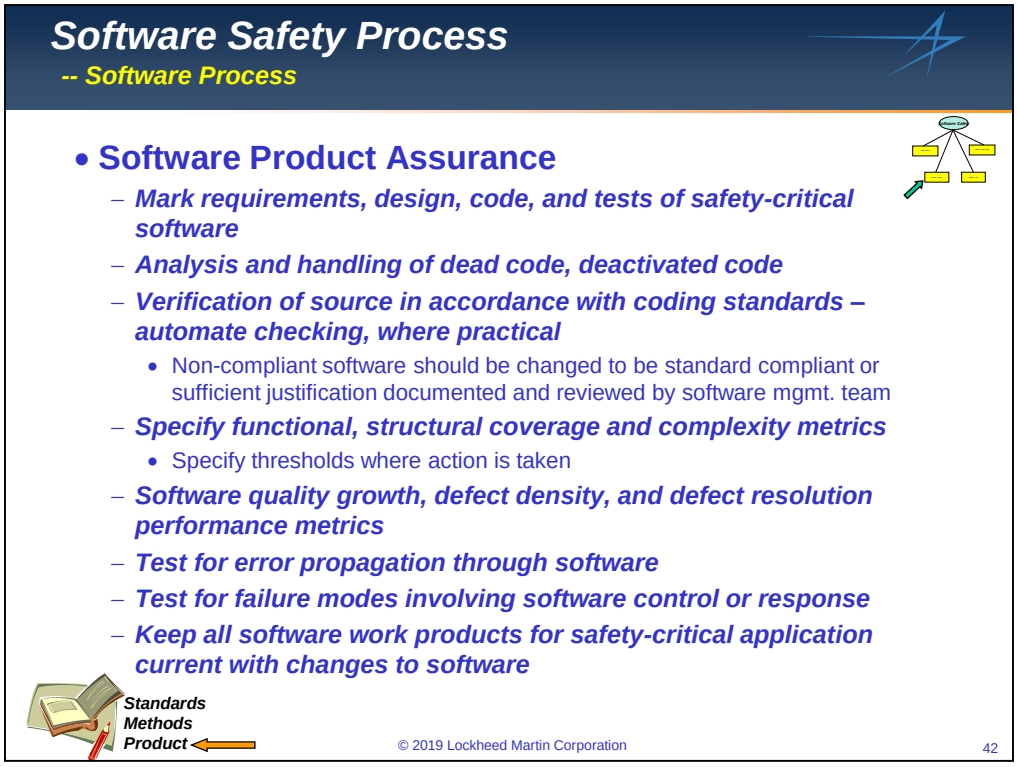
Standards
Methods ←
Product

Software Safety Process

Software Process:

Software Methods

- Bi-directional requirements Traceability to design to code to test procedures and back
- Defect prevention (used to eliminate whole classes of problems)
- Decision management for reuse and readiness
- Joint reviews
- Prototypes management
- Testing
- Software Product Reviews
- Reuse
- Change Impact Analysis

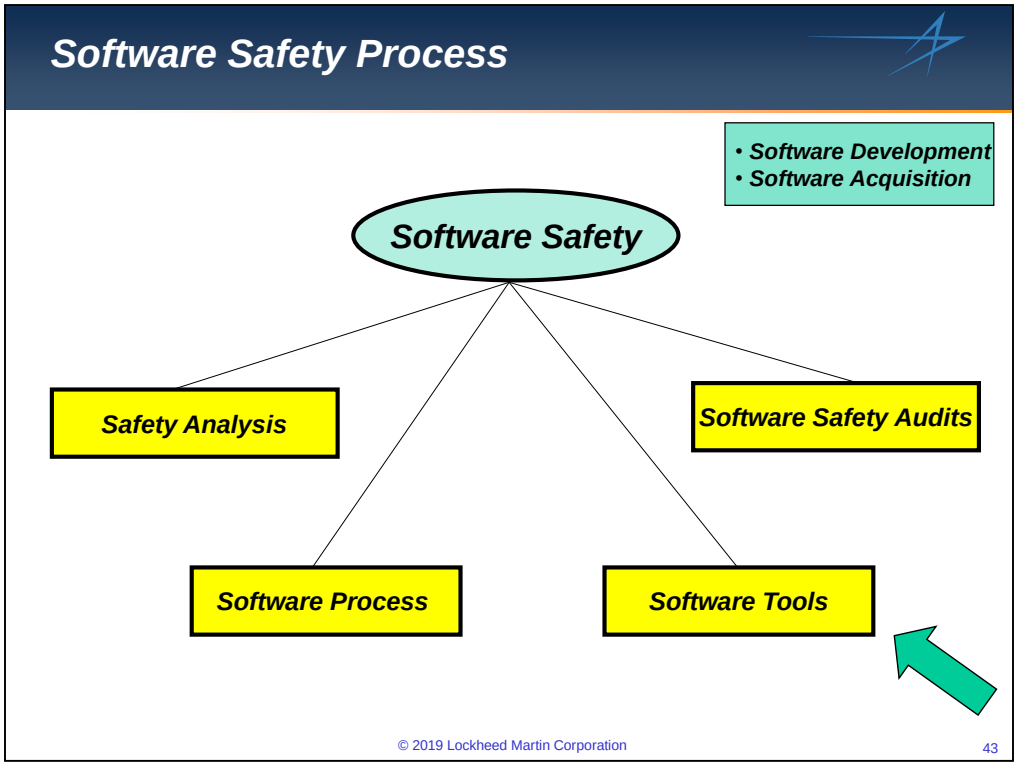


Software Safety Process

Software Process:

Software Product Assurance

- Mark work product elements as Safety-critical
- Handle dead code, deactivated code
- Check code against coding standards
- Specify and measure functional and structural coverage and manage complexity
- Software quality growth, defect density, and defect resolution metrics required
- Test Error propagation; failure modes
- Keep work products current with project.



Software Safety Process

Software Tools . . .

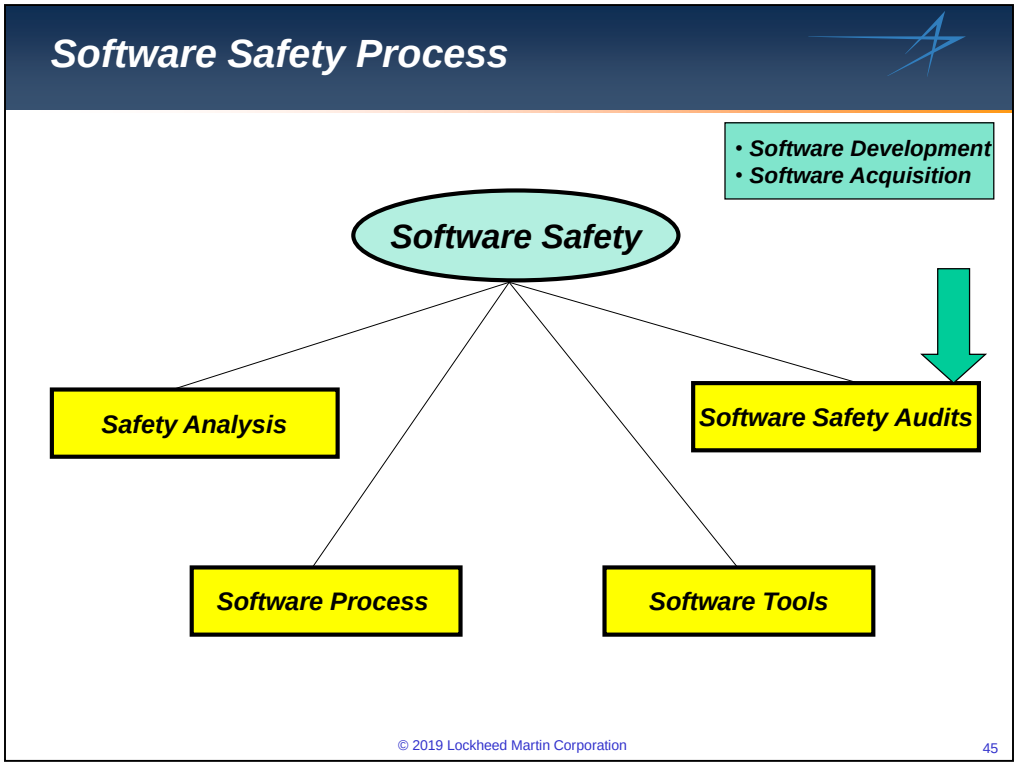
Software Safety Process
-- Software Tools

- **Software Tools (also captured in SDP)**
  - ***Configuration identification and control for key software tools used for safety-critical software***
    - Modeling tools that generate code
    - Build tools, utilities that construct executables
    - Analysis and debug tools used to test and report
  - ***Perform problem reporting and corrective action processing on key tools***
  - ***Qualification and re-qualification methods/approach for key tools and library usages. For example . . .***
    - *Tool vendor assumes all responsibility*
    - *Software team qualifies tools using documented test procedures; regression testing used where applicable*
    - *Software team conducts inspections of tool generated output to ensure tool is translating user input as designed; samples may be used*
    - *Software team partners with tool; vendor to mature key tools to company needs during program; vendor on contract to support work and agreed to changes*

Software Safety Process

Software Tools:

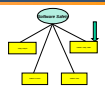
- Perform CM Identification and Control for key software tools, utilities
- Problem reporting & corrective action for key tools
- Qualify and re-qualify key tools for use



Software Safety Process

Software Safety Audits . . .

Software Safety Process
-- Software Safety Audits



- **Software Safety Audits (also in SDP)**
  - ***Auditing provides some assurance for acquirer that contractors have built what they intended to build and it is of required quality***
  - ***Audits usually accomplished through sampled reviews of process work products***
    - Variability in reviews dependant on auditor
  - ***Software Development Plan identifies and describes software process, including process details for safety-critical software***
  - ***Audit checks actual practice against written plans***
    - "*Say what you do*"
    - "*Do what you say*"

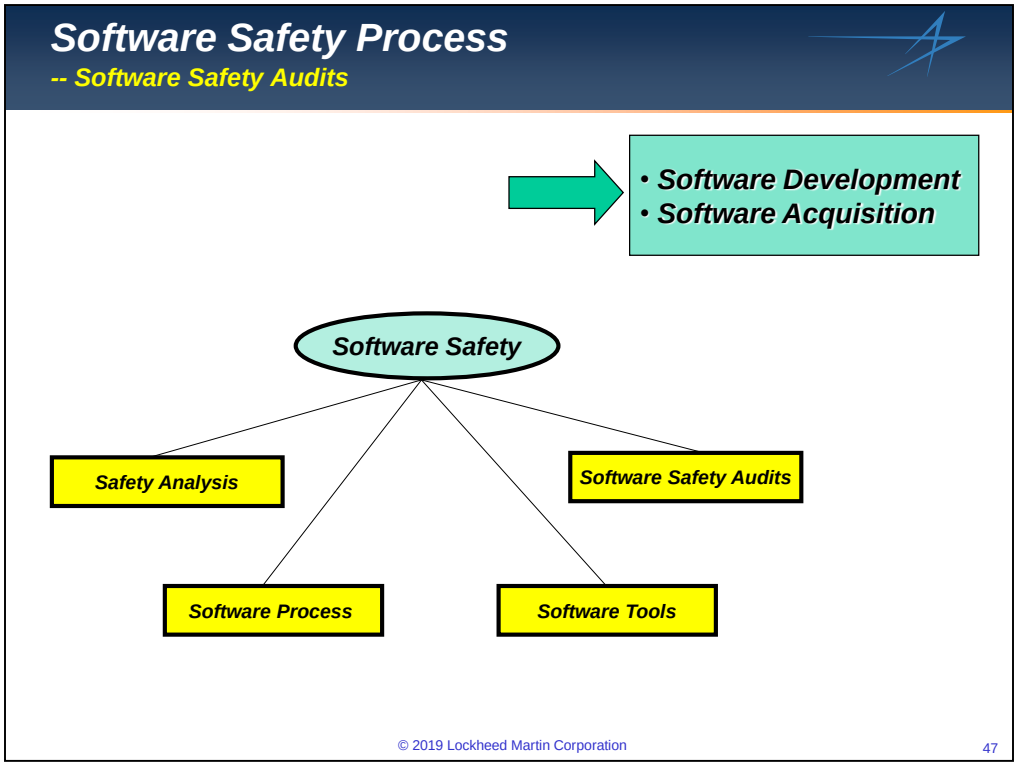
46

Software Safety Process

Software Safety Audits:

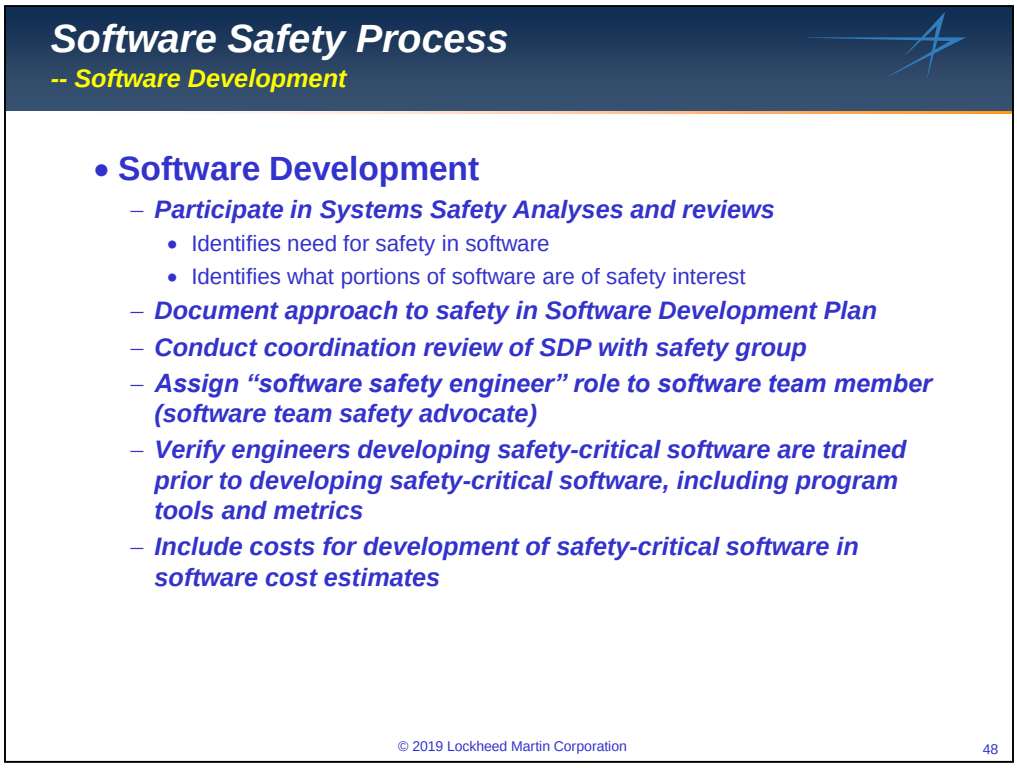
There will be audits (safety items will be part of it)

- Say what you do (SDP) and do what you say (software project work products)!



Software Safety Process

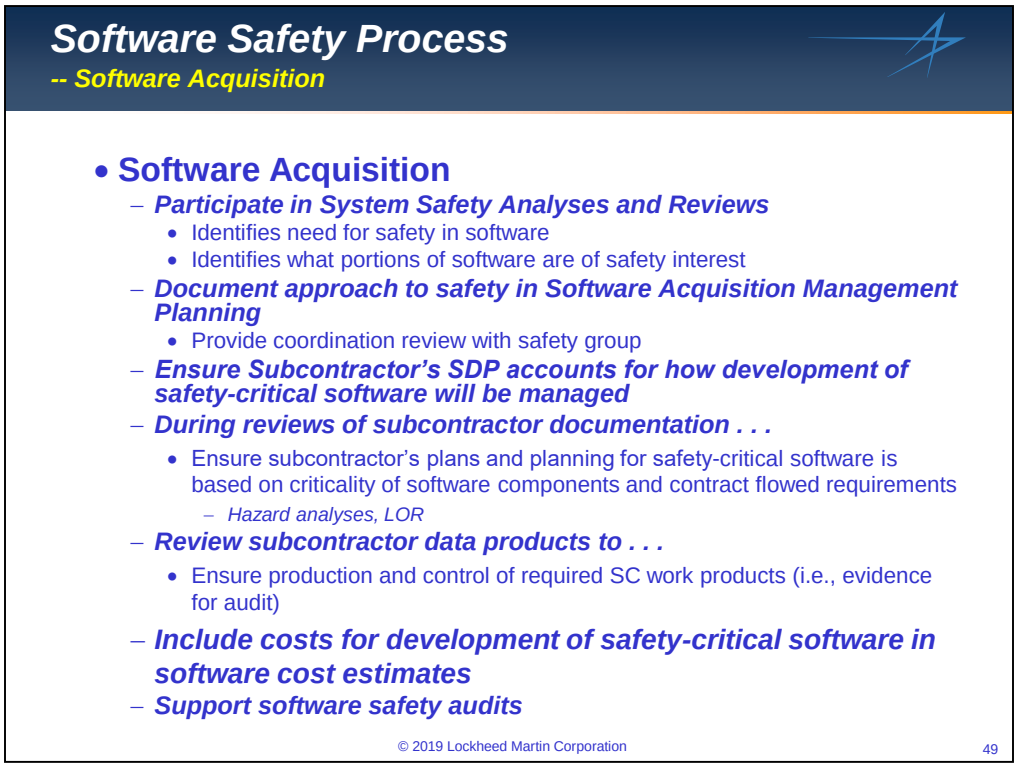
Software Development, Software Acquisition summary charts for safety in software . . .



Software Safety Process

Software Development:

In brief, this is at a high level the software engineering tasks for Safety. The process and details are captured in the SDP and the SW development environment and tools.



Software Safety Process

Software Acquisition:

In brief, this is at a high level and identifies software engineering tasks for Safety and Software Acquisition Management. The process and details are captured in the plans. Periodic reviews and collaborations with the subcontractor ensure requirements of software and safety are included in the work plans and products.

Whew!

- *"Sure sounds like a lot of requirements for building safety-critical software !"*
- **Software Engineering responds with risk reduction techniques to identified hazards and safety requirements through combination of . . .**
  - *Software Requirements Analysis and Design Choices*
  - *Software Process and Methods Choices*
  - *Tooling Choices and Management*
  - *Software Product Assurance and Audit*

- **Project must choose *balanced approach* to software safety based on requirements and sound engineering and economic business practice**

Software Safety Process

A quick summary of what is expected . . .

Software process requirements for safety are tailorable to the project and customer application needs . . .

Tailoring guidance for level of rigor provided by mapping the SW safety requirements to LoR.

But wait . . . That's not all ! !

- **For highest levels of software assurance, may also require . . .**
  - ***Independence in verification activities***
  - ***Testing of every decision structure, every condition shown to take all possible outcomes at least once and each condition shown to affect outcome independently (MC/DC)***
  - ***Source to Object Correspondence***
    - Used when highest assurance required and compiler generates object code not directly traceable to source

RTCA/DO-178B
SW Safety Levels
A. Catastrophic
B. Hazardous
C. Major
D. Minor
E. No Effect

- **When "system certification" is required by an independent certifying authority . . .**
  - ***Provides for independent oversight, collaboration, and verification***

Software Safety Process

For additional assurance requirements, found in DO-178 Level A process . . .

Independence in Verification – people other than developers or sometimes other than contractor test/review product

MC/DC – now assisted with tools . . .

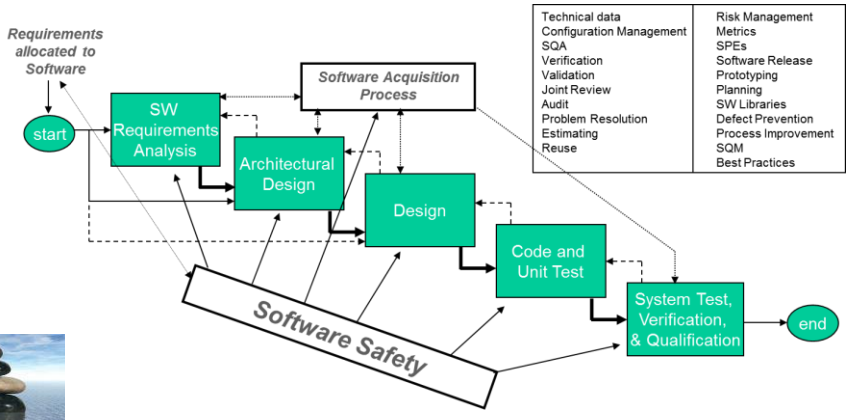
Source to Object code Correspondence: (FAA CAST-12, Position Paper)

. . . for Level A software for which structural coverage is performed on the source code, source code to object code traceability must be addressed (see paragraph 6.4.4.2b of DO-178B/ED-12B). Then, if the compiler generates object code that is not directly traceable to the source code, the applicant must identify that untraceable, compiler-generated object code and verify it.

-- EX: Compiler may create Initialization code, built in error detection, exception handling. Also aggressive optimization may eliminate instructions or functions, reorder instructions, . . . making it difficult to map source code to generated object.

Ultimately . . .

- **Project engineers must choose balanced approach to software safety based on system requirements and sound engineering and economic practice**
  - ***Checklists suggested with implementation based on criticality***



52

Software Safety Process

There is a lot of process requirements for engineering large software systems and with requirements for system safety. It is essential to choose a balanced approach to software safety choosing between sound engineering processes and economic practices (i.e., tools, process details, process compliance evidence requirements).

Tailoring Guidance Example

Software Safety Process

Tailoring Guidelines

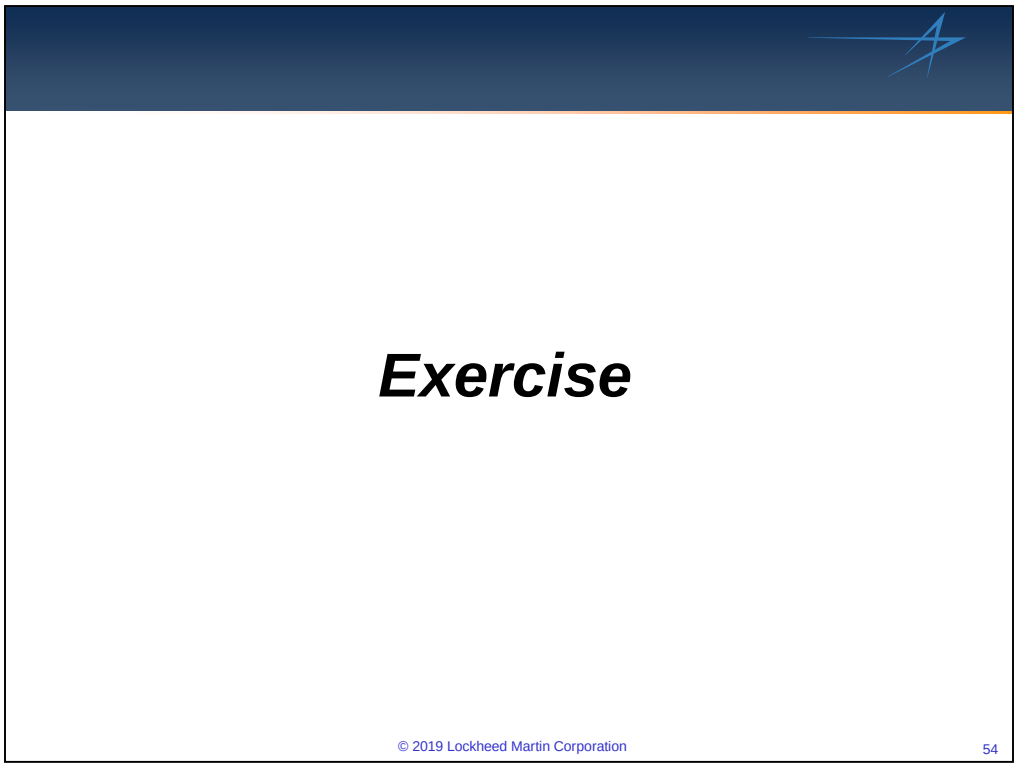
Reqt ID	Software Safety Practice Requirement	SWCI 1 Safety Critical	SWCI 2 Safety Significant	SWCI 3 Safety Related **
1	Identify the program safety levels of software with safety impact.	X	X	X
2	Identify and/or reference the software components associated with each program safety level.	X	X	X
3	Verify that software engineers have attended required software safety training courses prior to developing software with safety impact.	X	X	X
4	Establish a project process for enabling decisions regarding use, reuse, and readiness of software components with safety impact.	X	X	X
5	Identify and document constraints of architectural partitioning, processing and/or resource requirements, tools, software development methods or approaches, and/or specific documentation methods on the software development activities related to software with safety impact.	X	X	
6	Identify or reference standards (not a reference to a tool) for requirements development and for software design that specify the vocabulary, standards, and usages of software requirements and design methods, representations, and techniques.	X	X	
7	Specify or reference defect prevention activities for software with safety impact. These defect prevention activities will apply the approach documented in Section 4.16, Causal Analysis and Preventive Action.	X	X	X

*** SWCI 4 and 5 are already integrated into standard software process activities*

Software Safety

Tailoring guidance provided in software safety practice using example projects . . .

X -- means project must implement that requirement for that level of software being developed.



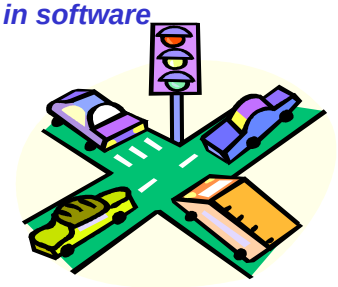
Course Exercise

This course exercise reinforces learning topics introduced earlier in the courseware.

The exercise background and introduction take about 15 minutes to introduce and about 30 to 45 minutes for the students to complete.

Exercise

- **Real-world problem to understand application of software safety**
  - ***4-way Traffic Light at intersection of high-speed highways***
- **Exercise is to examine design of traffic light system, determine if software is safety-critical, and if so . . .**
  - ***Identify the levels of criticality and why***
  - ***Modify software development and/or acquisition processes to lower safety risk in software***
  - ***Report findings***



55

Course Exercise

Traffic light exercise:

High speed (70 MPH on approach) rural highway

Cooperative electric company providing power

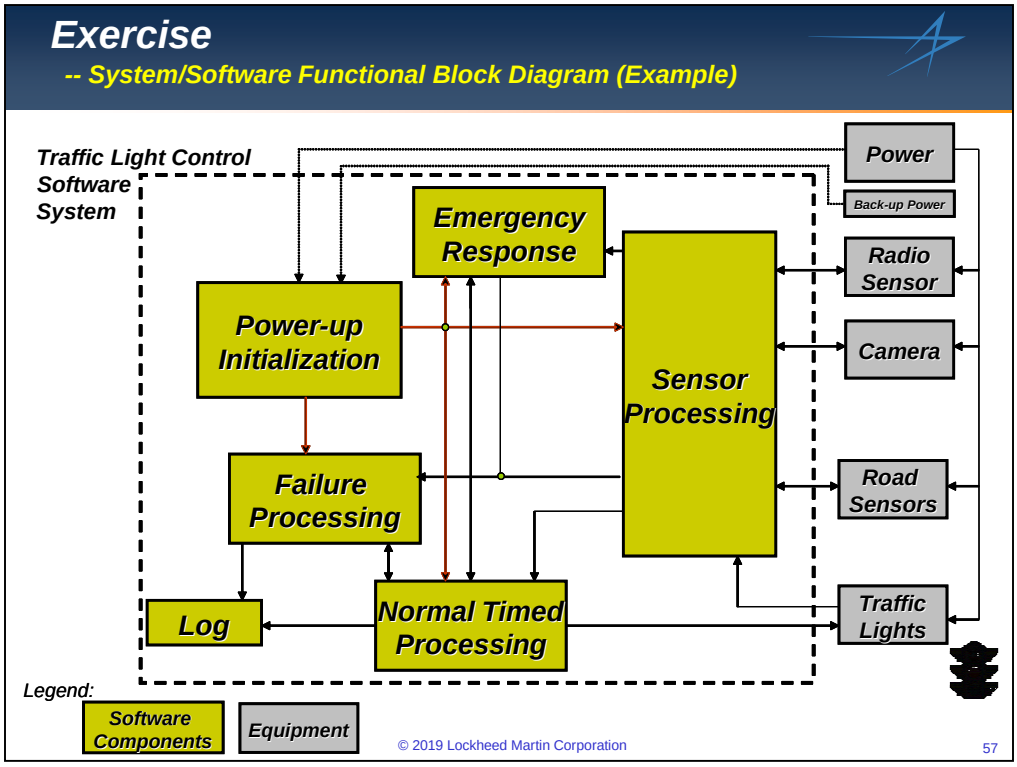
Lighted roadway at night in all directions; separate electrical feed for lights

Exercise
-- Requirements (Example)

- **Requirements *(Partial List)***
  - ***When power is first applied or restored, initialization processing will provide for orderly startup of traffic system computing resources***
  - ***During startup, traffic system will initialize lights to 4-way blinking red and wait for timed sequence instructions***
  - ***Once initialized, timed traffic light sequence will begin timed traffic light sequencing operation on N-S highway first***
  - ***Timed sequence may be shortened or lengthened based on in-road sensor processing requirements specified elsewhere***
  - ***4-way red lamps "on" condition will be initiated when correct signal is received from fire, ambulance, or police approaching intersection from any of 4 directions. Once activated, sequence will proceed for 5 seconds, then if another correct signal is not received within 2 seconds of deactivation, timed signal sequence will begin again on N-S highway first after 5 seconds has expired***
  - ***Unallowed lamp conditions:***
    - 4-way green on
    - 4-way amber on
    - 2-way green on with 2-way amber on
  - ***Back-up power shall be able to run traffic light signals continuously for 48 hours***
  - ***Intersection shall be illuminated during evening hours on each approach to traffic light and lighting power will be supplied by separate independent electrical feed . . .***
  - ***Etc. . . .***

Course Exercise

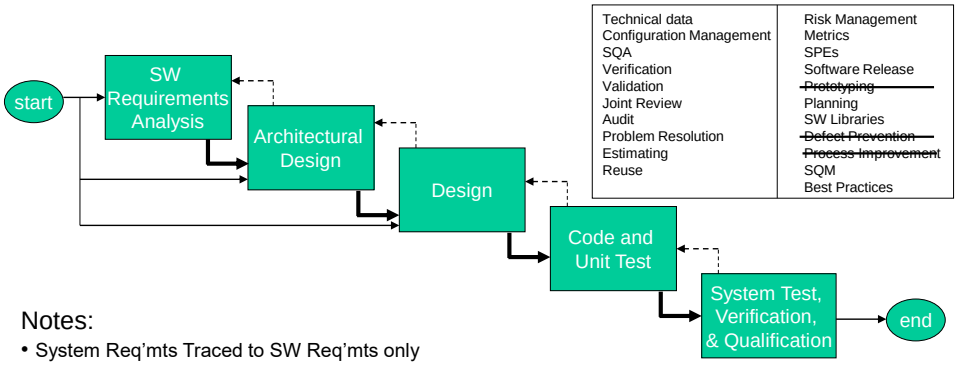
Partial list of requirements



Course Exercise

Functional Block diagram of traffic light system. Note hardware elements, software elements and what they do.

Exercise
-- Current Software Process (Example)



Notes:

- System Req'mts Traced to SW Req'mts only
- SW Req'mts, design, code, & test artifacts all electronic in tools; need specific tools to access
- Only informal peer reviews planned as cost reduction measure
- Characterization:
  - 20K Changed Lines of Code (logical) job estimate (adding emergency mode and associated failure processing, updating other components as necessary);
  - Total new size projection 70K SLOC Logical
  - OO, C++
  - COTS Operating System
  - 12 months to define, develop, certify, and deploy
  - DPS is certifying authority

Course Exercise

The current traffic light Software process.

Exercise
-- Safety Critical Functions (Example)

- **System Safety Engineering has determined following Functions are *Safety-Critical Functions*:**
  - *Display proper traffic lighting patterns for safe control of four-way highway traffic*
  - *Display proper sequence of red, amber, and green lights during normal traffic signal processing*
  - *Display lighting in proper timing of sequence of red, amber, and green lights during normal traffic signal processing*
  - *When system has entered a failure processing mode, display proper lighting sequence to notify traffic of intersection hazard*
  - *. . . . more . . . .*
- **Design Constraints:**
  - *System shall only allow 2 green lights to occur simultaneously, for through traffic lanes only*
  - *Length of amber lights being "on" shall be no more than 5 seconds and no less than 3.5 second*
  - *Failure mode of traffic signal shall be flashing red lamps in N-S direction and flashing amber lamps in E-W direction when power is available with system failure present*
  - *. . . . more . . . .*

Course Exercise

Safety-critical Functions . . .

Design Constraints . . . .

Exercise
-- Hazard Form (example)

Hazard Analysis Record

Hazard No. 001 **Engineer:** <name>	Project:: SW Safety Course System: Traffic Light Example Subsystem: Power Subsystem Phase:	Effectively: Initial Risk: Severity: __ Probability: __ Category: __ Modified Risk: Severity: __ Probability: __ Category: __	Date Opened: Status: Open ◯ In-Work ◯ FF Ready ◯ Monitored ◯

Description: If the power back-up equipment is unavailable and an interruption to electrical service occurs, the high-speed highway traffic light will be inoperative. Back-up power is only checked upon system startup.

Cause: The high-speed highway traffic light receives electrical power from the electric utility cooperative of the area. Power interruption is possible during electrical storms, grid outages, transmission line failure, and/or substation or transmission line equipment failure. During these events, electrical power may be unavailable to the traffic signal from seconds to hours depending on the circumstances of the event.

Effect: Probability of serious or fatal collision.

Requirements:

Controls:

Effects after Controls:

Remarks:

Hazard Closure Evidence:

Actions Remaining:

Review History:

Notes:

Course Exercise

Hazard form – maintained in the system safety hazard database. This is just a sample input form that can be printed as a database report.

This is the description of the hazard to be studied and mitigated. Discuss the details with class.

Course Exercise

-- Determining Criticality . . .

Risk Assessment Matrix				
Severity / Probability	Catastrophic (1)	Critical (2)	Marginal (3)	Negligible (4)
Frequent (A)	High	High	Serious	Medium
Probable (B)	High	High	Serious	Medium
Occassional (C)	High	Serious	Medium	Low
Remote (D)	Serious	Medium	Medium	Low
Improbable (E)	Medium	Medium	Medium	Low
Eliminated (F)	Eliminated			

Course Exercise

How to determine criticality . . .

Course Exercise

-- Determining Software Criticality . . .

Software Control Categories			
Level	**Name**	**Description**	**Considerations**
1	Autonomous (AT)	SW functionality that exercises autonomous control authority over potentially safety-significant HW systems, subsystems, or components without possibility of predetermined safe detection and intervention by a control entity to preclude occurrence of the mishap or hazard.	• Failure directly results in a mishap • No possibility of operator action to prevent the mishap.
2	Semi-Autonomous (SAT)	1. SW functionality that exercises control authority over potentially safety-significant HW systems, subsystems, or components allowing time for predetermined safe detection and intervention by independent safety mechanisms to mitigate or control the mishap or hazard. 2. SW item that displays safety-significant information requiring immediate operator entity to execute predetermined action for mitigation or control over the mishap or hazard. SW exception, failure, fault, or delay will allow, or fail to prevent, mishap occurrence.	• Failure could directly result in mishap if operator does not act • There is time for predetermined safe detection and intervention by independent safety mechanisms to mitigate or control the mishap
3	Redundant Fault Tolerant (RFT)	1. SW functionality that issues commands over safety-significant HW systems, subsystems, or components requiring a control entity to complete the command function. The system detection and functional reaction includes redundant, independent fault tolerant mechanisms for each defined hazardous condition. 2. SW that generates information of a safety-critical nature used to make critical decisions. The system includes several redundant, independent fault tolerant mechanisms for each hazardous condition, detection, and display.	• System detection and functional reaction includes redundant, independent fault tolerant mechanisms for each defined hazardous condition • SW with a failure condition requires another independent fault to result in a mishap
4	Influential	SW generates information of a safety-related nature used to make decisions by the operator, but does not require operator action to avoid a mishap.	• SW with a failure condition that reduces redundancy or safety margins but at least one independent mechanism remains to preclude a mishap • Operator makes the decisions
5	No Safety Impact (NSI)	SW functionality that does not possess command or control authority over safety-significant HW systems, subsystems, or components and does not provide safety-significant information. SW does not provide safety-significant or time-sensitive data or information that requires control entity interaction. SW does not transport or resolve communication of a safety-significant or time sensitive nature.	• After a SW failure there still are at least two independent mechanisms to preclude a mishap

- **Review SCC descriptions and select best match to situation**

Link to Block Diagram

Course Exercise

Determine which level of criticality applies . . . .

Course Exercise

-- Determining Software Criticality . . .

Software Safety Criticality Matrix				
SW Control Category	Severity Category			
	Catastrophic (1)	Critical (2)	Marginal (3)	Negligible (4)
1	SWCI 1	SWCI 1	SWCI 3	SWCI 4
2	SWCI 1	SWCI 2	SWCI 3	SWCI 4
3	SWCI 2	SWCI 3	SWCI 4	SWCI 4
4	SWCI 3	SWCI 4	SWCI 4	SWCI 4
5	SWCI 5	SWCI 5	SWCI 5	SWCI 5

SWCI	Level of Rigor Tasks
SWCI 1	Program shall perform analysis of requirements, architecture, design, and code and conduct in-depth safety-specific testing.
SWCI 2	Program shall perform analysis of requirements, architecture, design, and conduct in-depth safety-specific testing.
SWCI 3	Program shall perform analysis of requirements and architecture and conduct in-depth safety-specific testing.
SWCI 4	Program shall conduct safety-specific testing.
SWCI 5	Once assessed by safety engineering as Not Safety, then no safety specific analysis or verification is required.

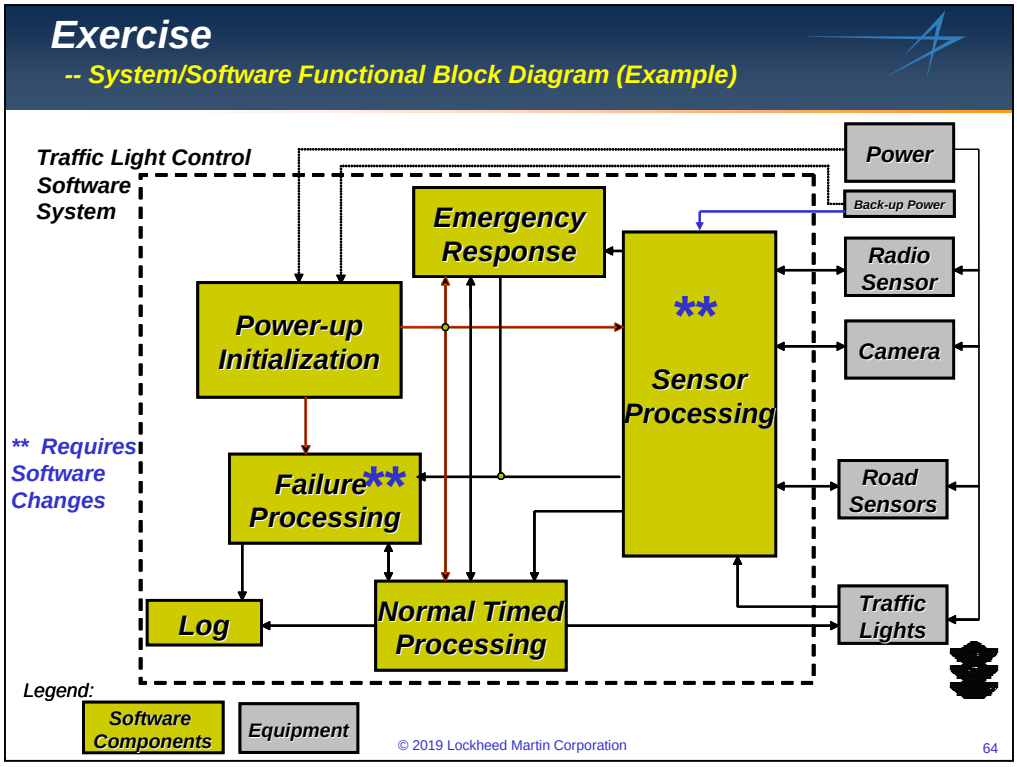
LoR

- **Map SCC with Severity Category to determine SWCI, which determine software level of rigor**

63

Course Exercise

Determine which SWCI number applies based om SCC and Severity criticality . . . .



Course Exercise

What components are affected by the proposed change . . .

Course Exercise
-- Software Safety Process Tailoring

Software Safety Process
Tailoring Guidelines

Reqt ID	Software Safety Practice Requirement	SWCI 1 Safety Critical	SWCI 2 Safety Significant	SWCI 3 Safety Related	**
1	Identify the program safety levels of software with safety impact.	X	X	X	
2	Identify and/or reference the software components associated with each program safety level.	X	X	X	
3	Verify that software engineers have attended required software safety training courses prior to developing software with safety impact.	X	X	X	
4	Establish a project process for enabling decisions regarding use, reuse, and readiness of software components with safety impact.	X	X	X	
5	Identify and document constraints of architectural partitioning, processing and/or resource requirements, tools, software development methods or approaches, and/or specific documentation methods on the software development activities related to software with safety impact.	X	X		
6	Identify or reference standards (not a reference to a tool) for requirements development and for software design that specify the vocabulary, standards, and usages of software requirements and design methods, representations, and techniques.	X	X		
7	Specify or reference defect prevention activities for software with safety impact. These defect prevention activities will apply the approach documented in Section 4.16, Causal Analysis and Preventive Action.	X	X	X	

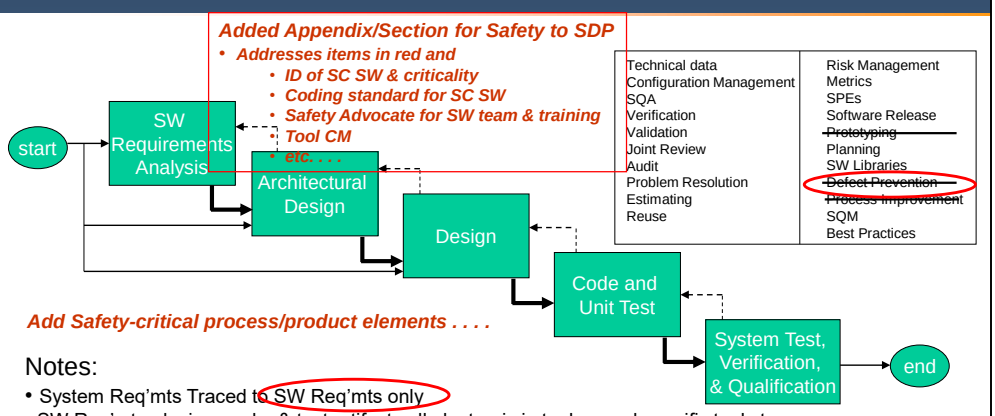
** SWCI 4 and 5 are already integrated as part of PM-4001

Course Exercise

How to determine a first pass of the software process requirements for the proposed change . . . .

Exercise

-- Software Process with Changes for Safety (Example)



Notes:

- System Req'mts Traced to SW Req'mts only
- SW Req'mts, design, code, & test artifacts all electronic in tools; need specific tools to access
- Only informal peer reviews planned as cost reduction measure
- Characterization:
  - 20K Changed Lines of Code (logical) job estimate (adding emergency mode and associated failure processing, updating other components as necessary);
  - Total new size projection 70K SLOC Logical
  - OO, C++
  - COTS Operating System
  - 12 months to define, develop, certify, and deploy
  - DPS is certifying authority

Course Exercise

Changes that affect the current software process . . . .

Exercise
-- Hazard Form (example)

Hazard Analysis Record

Hazard No. 001 **Engineer:** <name>	Project:: SW Safety Course System: Traffic Light Example Subsystem: Power Subsystem Phase:	Effectively: Initial Risk: Severity: __ Probability: __ Category: __ Modified Risk: Severity: __ Probability: __ Category: __	Date Opened: Status: Open ◯ In-Work ◯ FF Ready ◯ Monitored ◯

Description: If the power back-up equipment is unavailable and an interruption to electrical service occurs, the high-speed highway traffic light will be inoperative.

Cause: The high-speed highway traffic light receives electrical power from the electric utility cooperative of the area. Power interruption is possible during electrical storms, grid outages, transmission line failure, and/or substation or transmission line equipment failure. During these events, electrical power may be unavailable to the traffic signal from seconds to hours depending on the circumstances of the event.

Effect: Probability of serious or fatal collision.

Requirements: (Specification reference here.)

Controls: Design should provide monitor for back-up power and provide an indication to DOT when either back-up power is unavailable or insufficient to provide power to traffic light system continuously for a period of 48 hours. Software development process controls for developing function is SCC 2, SWCI 1.

Effects after Controls: Reduced occurrences of traffic light inoperative due to power or back-up power unavailability.

Remarks:

Hazard Closure Evidence: Test verification (e.g., in a test report) of this functional safety requirement for back-up power monitor.

Actions Remaining:

Review History:

Notes:

Course Exercise

What goes into the hazard database about the actions . . .

Course Exercise

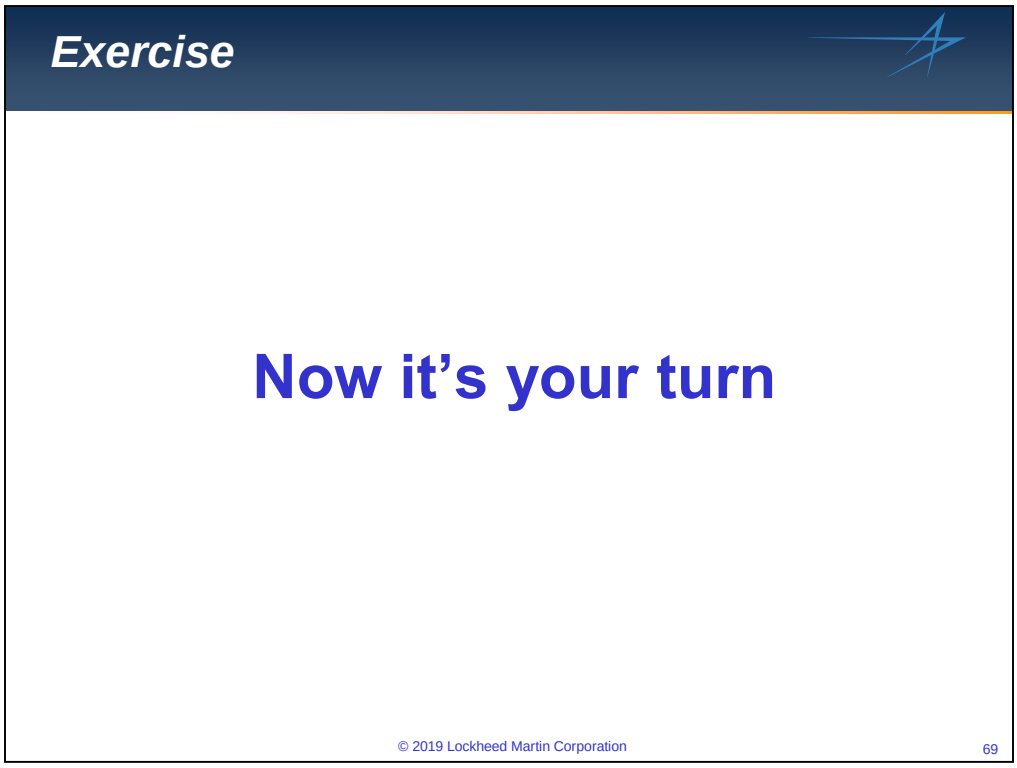
-- Determining Criticality After Controls . . .

Risk Assessment Matrix				
Severity / Probability	Catastrophic (1)	Critical (2)	Marginal (3)	Negligible (4)
Frequent (A)	High	High	Serious	Medium
Probable (B)	High	High	Serious	Medium
Occassional (C)	High	Serious	Medium	Low
Remote (D)	Serious	Medium	Medium	Low
Improbable (E)	Medium	Medium	Medium	Low
Eliminated (F)	Eliminated			

Course Exercise

With the hazard controls in place, what is the new HRI and why. . . .

Note that typically one move across the row (i.e., probably of occurrence) in the mitigation activity rather than down the column (i.e., changing the severity).



Course Exercise

Now students should have enough background to complete the exercise. Need to discuss their resources and hazards next before setting them off to work . . .

Course Exercise
-- Risk Assessment Matrix

Risk Assessment Matrix				
Severity / Probability	Catastrophic (1)	Critical (2)	Marginal (3)	Negligible (4)
Frequent (A)	High	High	Serious	Medium
Probable (B)	High	High	Serious	Medium
Occassional (C)	High	Serious	Medium	Low
Remote (D)	Serious	Medium	Medium	Low
Improbable (E)	Medium	Medium	Medium	Low
Eliminated (F)	Eliminated			

Course Exercise

Risk Assessment Matrix . . . .

Exercise
-- Hazard Form

Hazard Analysis Record

Hazard No. **Engineer:** <name>	Project:: SW Safety Course System: Traffic Light Example Subsystem: Power Subsystem Phase:	Effectively: Initial Risk: Severity: __ Probability: __ Category: __ Modified Risk: Severity: __ Probability: __ Category: __	Date Opened: Status: Open ◯ In-Work ◯ FF Ready ◯ Monitored ◯

Description:

Cause:

Effect:

Requirements: (Specification reference here.)

Controls:

Effects after Controls:

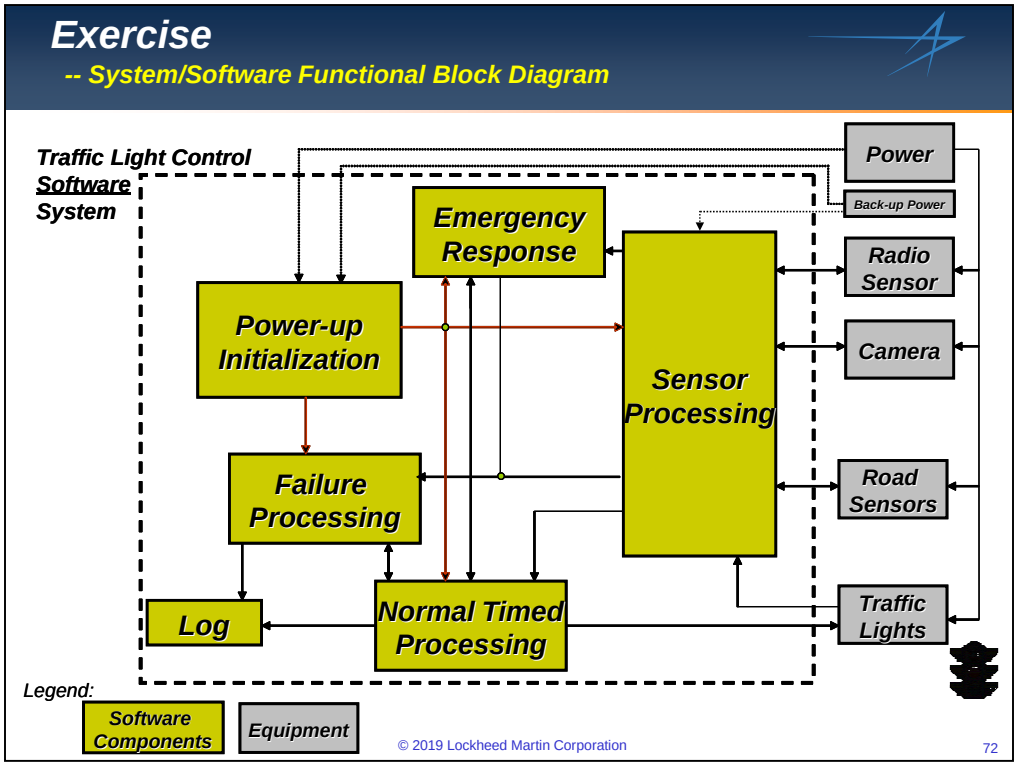
Remarks:

Hazard Closure Evidence:

71

Course Exercise

Hazard form to fill out . . . .

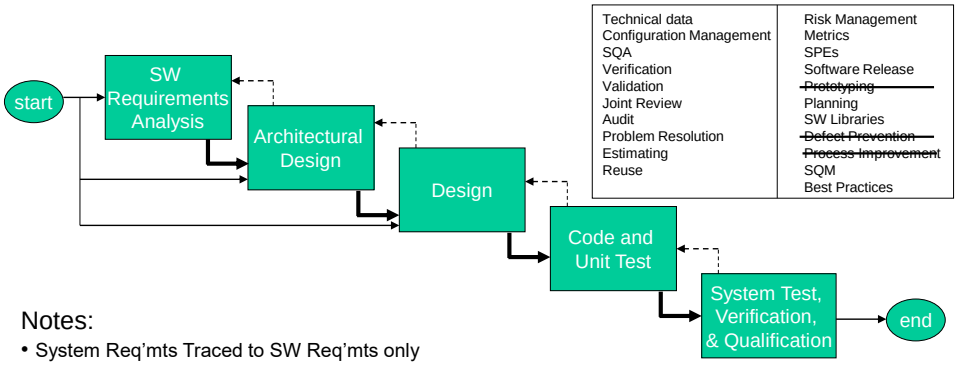


Course Exercise

Block diagram to use . . .

Exercise

-- "Current" Software Process



Notes:

- System Req'mts Traced to SW Req'mts only
- SW Req'mts, design, code, & test artifacts all electronic in tools; need specific tools to access
- Only informal peer reviews planned as cost reduction measure
- Characterization:
  - 20K Changed Lines of Code (logical) job estimate (adding emergency mode and associated failure processing, updating other components as necessary);
  - Total new size projection 70K SLOC Logical
  - OO, C++
  - COTS Operating System
  - 12 months to define, develop, certify, and deploy
  - DPS is certifying authority

Course Exercise

Current Software Process. Just mark up this page with summary or high-level software process changes . . . .

Course Exercise

-- Determining Software Criticality . . .

Software Control Categories			
Level	**Name**	**Description**	**Considerations**
1	Autonomous (AT)	SW functionality that exercises autonomous control authority over potentially safety-significant HW systems, subsystems, or components without possibility of predetermined safe detection and intervention by a control entity to preclude occurrence of the mishap or hazard.	• Failure directly results in a mishap • No possibility of operator action to prevent the mishap.
2	Semi-Autonomous (SAT)	1. SW functionality that exercises control authority over potentially safety-significant HW systems, subsystems, or components allowing time for predetermined safe detection and intervention by independent safety mechanisms to mitigate or control the mishap or hazard. 2. SW item that displays safety-significant information requiring immediate operator entity to execute predetermined action for mitigation or control over the mishap or hazard. SW exception, failure, fault, or delay will allow, or fail to prevent, mishap occurrence.	• Failure could directly result in mishap if operator does not act • There is time for predetermined safe detection and intervention by independent safety mechanisms to mitigate or control the mishap
3	Redundant Fault Tolerant (RFT)	1. SW functionality that issues commands over safety-significant HW systems, subsystems, or components requiring a control entity to complete the command function. The system detection and functional reaction includes redundant, independent fault tolerant mechanisms for each defined hazardous condition. 2. SW that generates information of a safety-critical nature used to make critical decisions. The system includes several redundant, independent fault tolerant mechanisms for each hazardous condition, detection, and display.	• System detection and functional reaction includes redundant, independent fault tolerant mechanisms for each defined hazardous condition • SW with a failure condition requires another independent fault to result in a mishap
4	Influential	SW generates information of a safety-related nature used to make decisions by the operator, but does not require operator action to avoid a mishap.	• SW with a failure condition that reduces redundancy or safety margins but at least one independent mechanism remains to preclude a mishap • Operator makes the decisions
5	No Safety Impact (NSI)	SW functionality that does not possess command or control authority over safety-significant HW systems, subsystems, or components and does not provide safety-significant information. SW does not provide safety-significant or time-sensitive data or information that requires control entity interaction. SW does not transport or resolve communication of a safety-significant or time sensitive nature.	• After a SW failure there still are at least two independent mechanisms to preclude a mishap

- **Select closest SCC to your hazard situation**

Link to Block Diagram

Course Exercise

Software Criticality Category . . . .

Course Exercise

-- Determining Software Criticality . . .

Software Safety Criticality Matrix				
SW Control Category	Severity Category			
	Catastrophic (1)	Critical (2)	Marginal (3)	Negligible (4)
1	SWCI 1	SWCI 1	SWCI 3	SWCI 4
2	SWCI 1	SWCI 2	SWCI 3	SWCI 4
3	SWCI 2	SWCI 3	SWCI 4	SWCI 4
4	SWCI 3	SWCI 4	SWCI 4	SWCI 4
5	SWCI 5	SWCI 5	SWCI 5	SWCI 5

SWCI	Level of Rigor Tasks
SWCI 1	Program shall perform analysis of requirements, architecture, design, and code and conduct in-depth safety-specific testing.
SWCI 2	Program shall perform analysis of requirements, architecture, design, and conduct in-depth safety-specific testing.
SWCI 3	Program shall perform analysis of requirements and architecture and conduct in-depth safety-specific testing.
SWCI 4	Program shall conduct safety-specific testing.
SWCI 5	Once assessed by safety engineering as Not Safety, then no safety specific analysis or verification is required.

- **Select SWCI that maps the SCC and Severity Category for your hazard situation**
- **This SWCI then identifies the Level of Rigor needed for your software development for the modification**
- **With these system changes, let's reassess using the hazard risk matrix (next page)**

75

Course Exercise

Current Software Criticality Matrix . . . Find you hazard SWCI # which identifies the LoR.

Course Exercise
-- Ending Risk Assessment Matrix

Risk Assessment Matrix				
Severity / Probability	Catastrophic (1)	Critical (2)	Marginal (3)	Negligible (4)
Frequent (A)	High	High	Serious	Medium
Probable (B)	High	High	Serious	Medium
Occassional (C)	High	Serious	Medium	Low
Remote (D)	Serious	Medium	Medium	Low
Improbable (E)	Medium	Medium	Medium	Low
Eliminated (F)	Eliminated			

Course Exercise

Re-evaluate the hazard risk now that controls have been designed and implemented. It should be less.

Exercise
-- Exercise Instructions

- **Exercise instructions**
  - ***Divide class into work groups***
  - ***Assignment:***
    - Document at least one hazard on hazard form provided
    - Determine the criticality of hazard *(use HRI table)*
    - Define approach to mitigate hazard
    - Identify which software engineering process requirements are relevant for software development of your assigned component (Use checklists provided); finish hazard control.
    - Each group reports results back to class
  - ***Use your best engineering judgment and rationale with information given (make assumptions as necessary and discuss in group)***
  - ***Assume software process is already documented but with nothing for safety***
    - Assume OO process, C++ IDE, Desktop test tools, CM, etc.

Course Exercise

Instructions . . .

Split the class into 5 or less working groups for the rest of the exercise . . . .

Some Ground Rules for Exercise

- **You are free to be as creative as you'd like with solutions**
  - ***Cost, budget, schedule are flexible, not constraints***
- **You may use redundant equipment but you must have at least one set of changes that affects software**
  - ***This is Software Safety***
- **You must provide solutions that reduce the hazard risk index except for . . .**
  - ***No tunnels or bridges around intersection***
  - ***No new concrete barriers or collision protection systems***
  - ***No other signage or lighting is needed at or near intersection***

Exercise Ground Rules

Some ground-rules to consider in doing this assignment:

- Blank check
- At least one software change is needed
- Propose solutions to reduce risk except for:
  - No tunnels
  - No barriers or collision protection systems
  - No other signage is needed.

Exercise

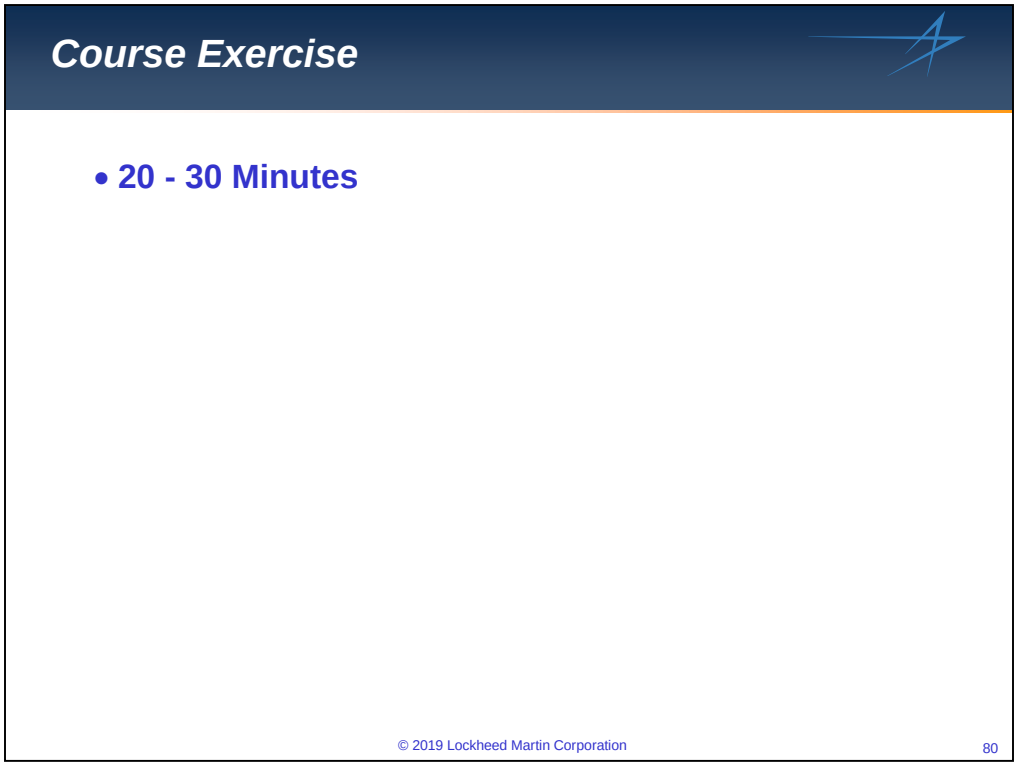
-- Exercise Hazards for Group Work

- **Red/(green) lamp burns out on the N-S bound lane leading to no stop/(go) indication for on-coming traffic that did not see the previous traffic light transition.**
- **[Barn swallows build a nest on the traffic light fixture (unnoticed?).] The RF sensor circuit [is compromised and] fails to engage all-stop emergency response mode for fire and rescue.**
- **Embedded roadway sensor circuit fails leading to traffic not being sensed for left-turn lane crossing traffic. Left turn sequence never engages.**
- **On routine maintenance run after a morning severe electrical storm, it was observed that battery back-up power was depleted but there was no message from the traffic light system. Traffic light was also observed to be in-operative. After rebooting system, message was generated; backup power was repaired.**
- **There is no way for traffic light to verify that it is sequencing lights properly or improperly during normal operation. It is possible for the traffic light to operate out-of-sequence and yet not report an error creating intersection hazard.**

79

Course Exercise

Review each of these Hazards and assign one to each of the work groups . . .



Course Exercise

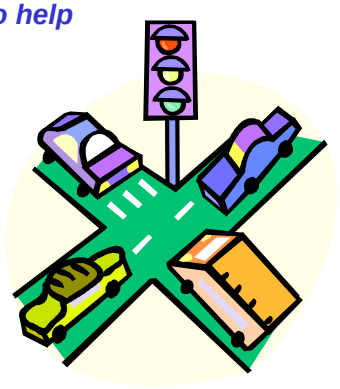
You have 20 to 30 minutes.

Note to instructor –

- Each team will take 2 to 3 minutes to present their solution
- It will take about 20 minutes to wrap-up the class after the exercise. You can do it a little faster, but not much.
- Adjust your instruction delivery timing as appropriate to get the class out on time.

Exercise Review

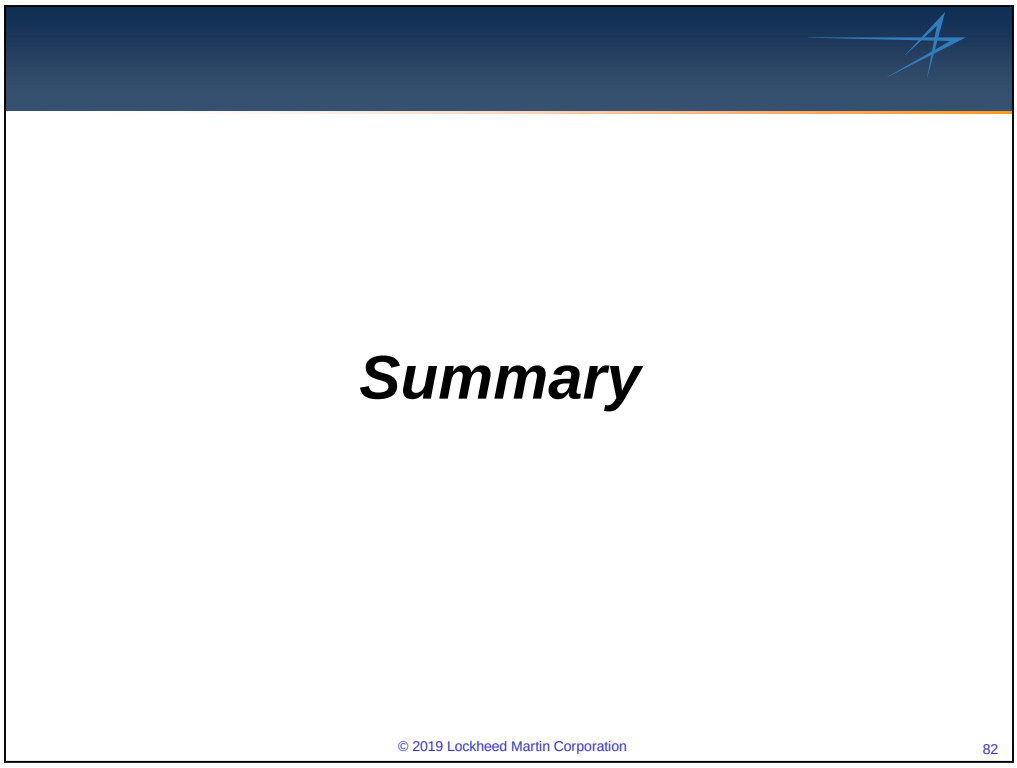
- **You were to examine design of traffic light system, define hazard and control, determine if software was safety critical, identify levels of criticality, and why and modify software process accordingly**
  - ***Checklists were provided to help***
- **Present solutions . . . .**



Course Exercise

Class returns from break to present results of their work.

Note that even though this was a simple exercise, this is the general process of discussing and implementing hazard mitigation in software systems. On larger systems, there can be hundreds and even thousands of hazards to work through . . .



Course Summary

Sums up the salient points for the audience in about 20 minutes and also provides some thinking points for a little later on . . . .

Summary

- **Safe Software ≠**
  - *Lower software defect rates*
  - *Reliable Software*
  - *Secure Software*
- **Safety is a systems attribute**
  - *Software Engineering and software are contributors to safe systems and safe operations*
- **Safety Engineering conducts hazard analysis on program**
  - *Software Engineering works with Safety Engineering to help identify and characterize hazards involving the command, control, and/or monitoring of critical functions necessary for safe operation of system*
- **Risk Consequences of Software Safety involve**
  - *People*
  - *Money*
  - *Environment*

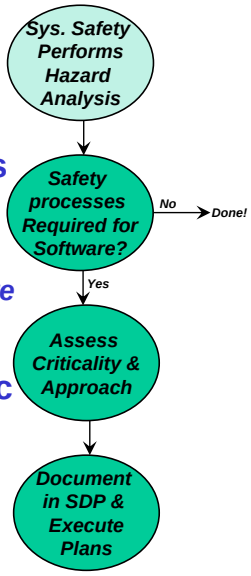
Course Summary

Safe software is about managing the risk of hazards, in complex software intensive systems, being raised due to the operation of software within the context of the operational system. This is software behaving badly. Hazard mitigation activities with respect to software help to lower the probable occurrence of these types of hazards being raised.

Note: Mishaps always involve a ‘chain of events’. The mishap is usually never related to single fault or failure.

Summary (Cont'd)

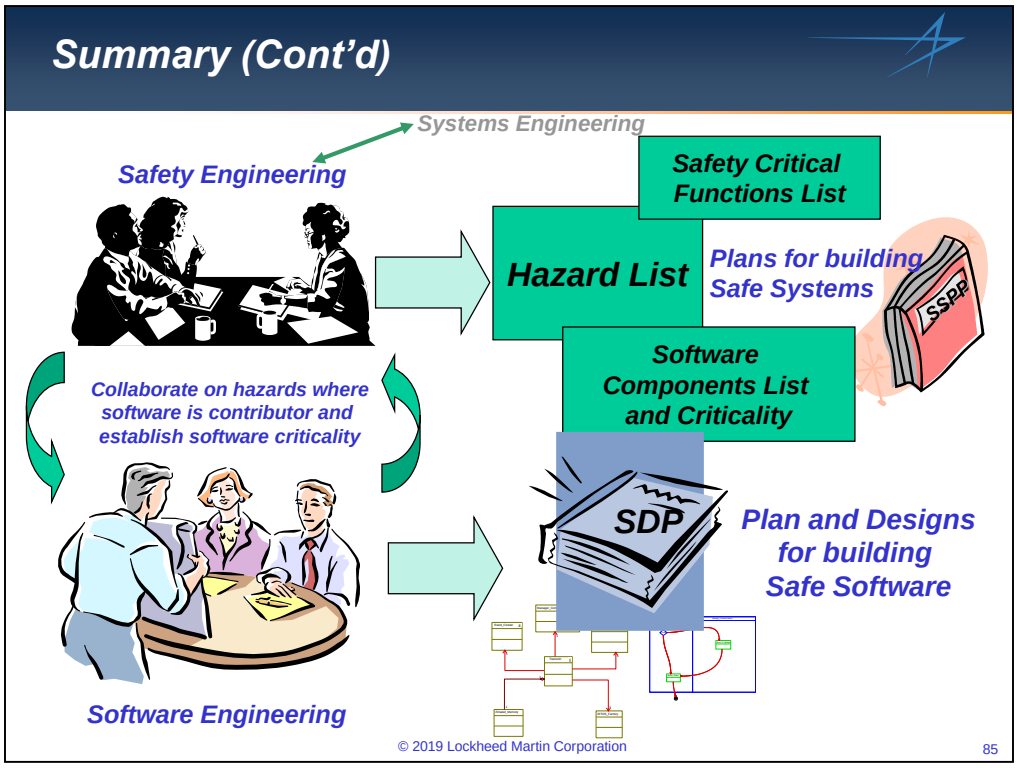
- **Safety processes in software apply for . . .**
  - *Developed software*
  - *Acquired software*
- **Software Development Process documents Software engineering and Software Safety practices**
  - *Provides context for developing product software*
  - *Software process requirements*
- **Software Safety process tailored to specific application**
  - *in Software Development Plan (SDP)*



Course Summary

4-step process . . .

Process should be tailored to fit the need.



Course Summary

Build chart . . .

Who does what . . .

Quick summary . . . 3rd time . . . just like Disneyland parking lot . . . .

Software Failures Affect US

… a few more recent examples and last reminders

- **Software Glitch Delayed Release of Results**
  - ***(2014) – New Brunswick, an 'off-the-shelf' computer program used by the voting machines failed and delayed vote tabulations by a day. Recount called, results accuracy questioned . . . Planning to use same system next time . . . .***
- **Ford Recalls F150 that could hit the Brake when not supposed to . . .**
  - ***Ford is recalling over 37,000 trucks because a software error can in the adaptive cruise control – when the pickup passes a highly reflective track, radar can be fooled that obstacle is in lane and then hit brakes, sound collision-warning system . . . .***

- **Prius Problems Traced to Software Glitch**
  - **June, 2015: *Toyota Motor Corp is recalling 625,000 cars due to a software problem in the popular hybrid Prius automobile after complaints that the gas-electric hybrid cars stall or shuts down without warning while driving . . . .***

Course Summary

Some more notable software failures . . . .

- Software delays release of election results in New Brunswick, Canada
- Ford recalls 37,0000 Ford F 150s due to unexpected behavior of adaptive cruise control system – anti-collision – brakes behavior – remember the self-driving car earlier?
- Still seeing problems with some Toyota Prius cars when the hybrid system gets shut down by the software while the owner is driving the car – expected shut down . . . . Imagine if this happened at highway speeds and you were being followed closely by an 18-wheeler at the time . . . .

Software Failures Affect US

… a few more examples and last reminders

- **Mishaps where software-related problems were reported to play a significant role . . .**

Year	Deaths	Description
1985	3	Therac-25 Software Design Flaw lead to radiation overdoses in treatment of cancer patients
1991	28	Software prevents Patriot missile battery from targeting SCUD missile. Hits army barracks
1995	159	AA jet crashes into mountain in Cali, Columbia. Software presented insufficient and conflicting information to pilots who got lost
1997	1	Software causes morphine pump to deliver lethal dose to patient
2000	4	Crash of V-22 Osprey tilt-rotor helicopter caused by software anomaly
2001	5	Panamanian cancer patients overdosed with radiation due to faulty software
2003	3	Software failure contributes to power outage across NW U.S. and Canada

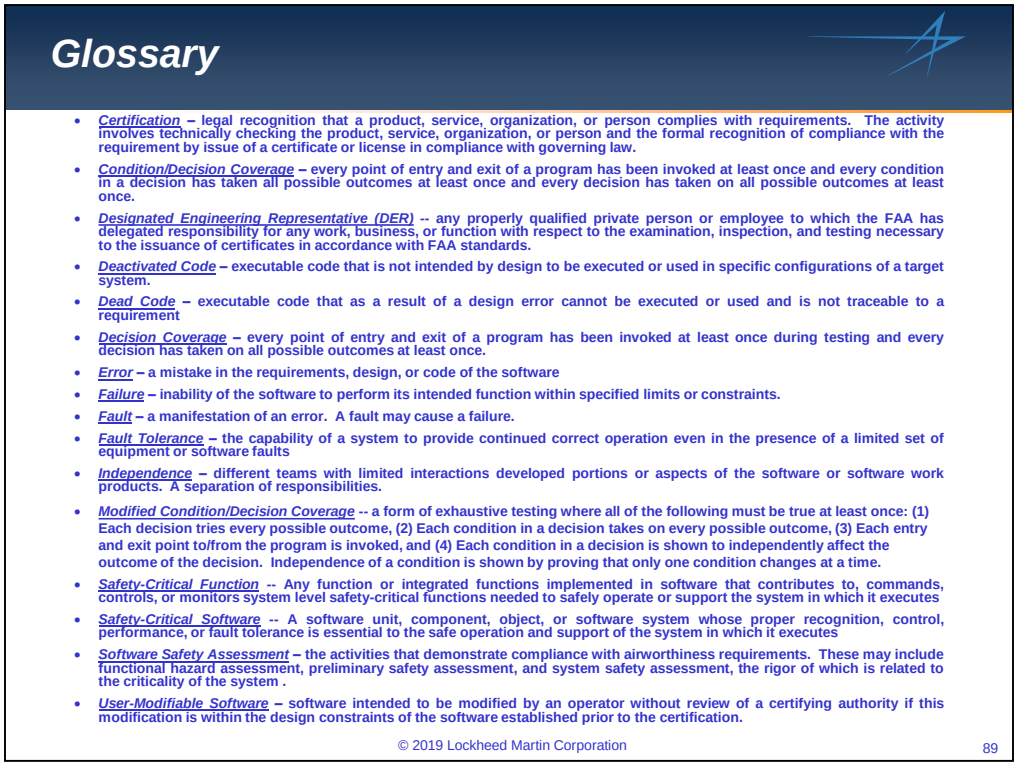
RE: Baseline Magazine, "Eight Fatal Software-Related Accidents", March 4, 2004

Course Summary

These are all mishaps that have killed people. In each of these events, software was identified as a significant contributing factor to the accident/mishap.

- Therac 25 – SW design flaws lead to radiation overdoses of some US and Canadian cancer patients – could be more . . . .
- Software accuracy error in system design – if system was run longer than 8 hours without a reboot, errors accumulate and system would lose track of missile
- Jury says maker of flight management system was 17% responsible for aa crash and death of passengers and crew – software presented insufficient and conflicting information to pilots, who got lost . . . .
- Software logic error cause lethal dosage if morphine delivered to patient – vendor reprograms device
- V-22 Ospry crash blamed on hydraulics line rupture and 'software anomaly' – pressing reset repeatedly caused aircraft to power up and power down engines eventually leading to stall.
- Panamanian doctors used 5 radiation shields instead of 4 over the patient to protect patient, but software was only designed to handle 4 blocks. 28 patients overdosed – 5 died due to radiation poisoning, others died but unclear if by cancer progression or the treatment. Doctors could be jailed on second degree manslaughter charges . . .
- Software alarm system software partially blamed for blackout.

Note: Each of the reported mishaps, failures reported in this entire presentation have been documented and are available on the internet with the provided URL links in this training material or by using keyword search.



Glossary

For those of you actually involved in the development of safety-critical software, this glossary provides a starting domain vocabulary.

Further Reading and References . . .

- Safeware: System Safety and Computers, Nancy Leveson
- Software System Safety Handbook, A Technical and Managerial Team Approach, Joint Services Computer Resources Management Group, U.S. Navy, and the U.S. Air Force.
- FAA System Safety Handbook, Appendix J: Software Safety
- NASA-STD-8719.13A – Software Safety
- *IEEE 1228 – IEEE Standard for Software Safety Plans*
- *EIA SEB6-A – System Safety Engineering in Software Development*
- *MIL-STD-882E – Standard Practice for System Safety*
- RTCA, Inc., *DO-178C, Software Considerations in Airborne Systems and Equipment Certification, and . . .*
  - *RTCA, Inc., DO-248C, Supporting Information for DO-178C and DO-278A*
  - *RTCA, Inc., DO-330, Software Tool Qualification Considerations*
  - *RTCA, Inc., DO-331, Model-Based Development and Verification Supplement to DO-178C and DO-278A*
  - *RTCA, Inc., DO-332, Object-Oriented Technology and Related Techniques Supplement to DO-178C and DO-278A*
  - *RTCA, Inc., DO-333, Formal Methods Supplement to DO-178C and DO-278A*
- The DACS Software Reliability Sourcebook, Data & Analysis Center for Software
- The System Safety Society
- International System Safety Conferences
- Graduate school courseware offerings in Software Safety
- Consultants courseware offerings in Software Safety
- And many more . . .

Further Reading

References for further reading . . . .

Your Instructor . . .

Dr. Michael F. Siok, PE, ESEP
Lockheed Martin Aeronautics Company
P.O. Box 748, MZ 5940
Fort Worth, TX 76101
Tel: (817) 777-4234
Email: Mike.F.Siok@lmco.com

Your instructor for this Software Safety Overview training course . . . .

Lockheed Martin Aeronautics Company

http://www.lockheedmartin.com/aeronautics/

Lockheed Martin Aeronautics Video . . .

Lockheed Martin Aeronautics Company

Lockheed Martin ‘go anywhere, do anything’ video