Lockheed Martin Aeronautics Company

Safety and Software

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

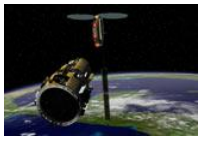
- **What is Safe Software, Software Safety ? ? ? ? ?**

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

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

… 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 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

Course Objectives
-- Software Safety

- **Introduce Need for *Safety in Software***
  - ***Requirements for safety and software at LM Aeronautics and Lockheed Martin Corporation***
  - ***Standards and Industry Practice***
  - ***Goal of Software Safety Program at Aero***
- **Provide an overview of a software engineering safety practice**
  - ***Software Safety Process***
  - ***General Tailoring Approach***
- **Reinforce principles and concepts with interesting group exercise**

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 (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 (Cont'd)

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

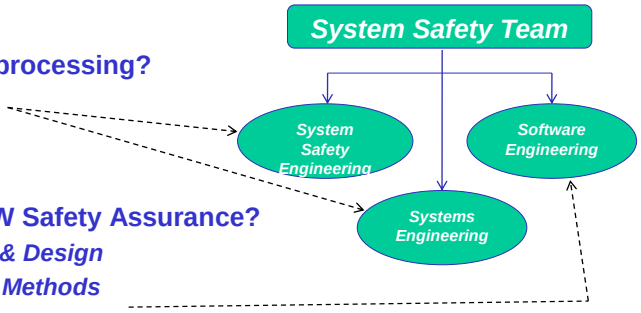
- **Customer requirements for safety usually specified in contracts**
  - ***E.g., MIL-STD 882, ARP-4761***
  - ***Software not excluded from safe systems operation***

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

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*.**

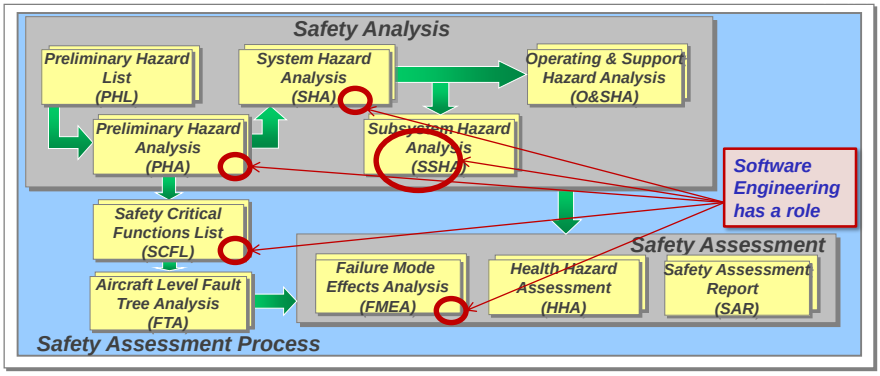
- **How to ID critical processing?**
  - ***Hazard Analysis***



- **How to Provide SW Safety Assurance?**
  - ***SW Architecture & Design***
  - ***SW Processes & Methods***
  - ***SW Tooling***

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.***



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 (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

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 (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) --**

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 (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 (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 (Cont'd)

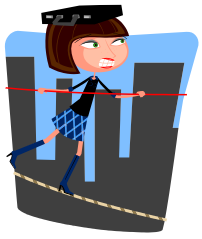
-- MIL-STD 882E SWCI, Risk, LOR, and Consequences

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 (Cont'd)

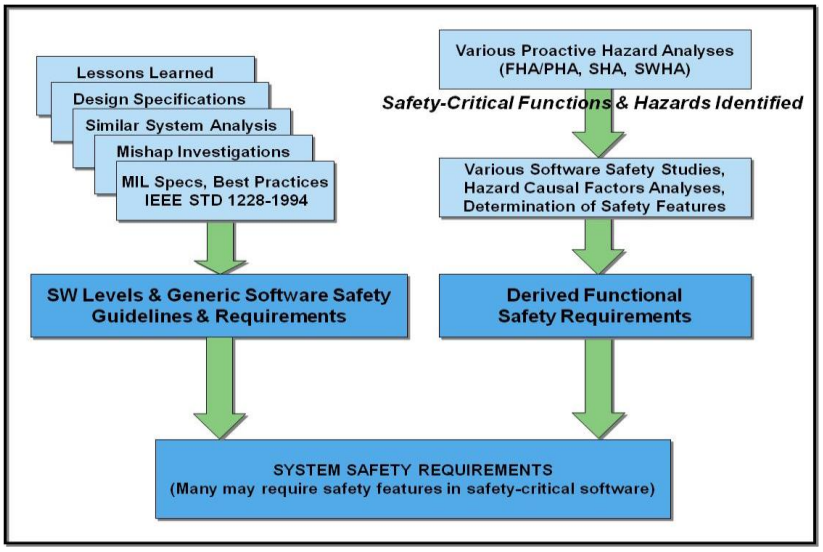
-- MIL-STD 882D Mishap Severity Categories (Cont'd)

- **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 (Cont'd)

-- From Hazards to Requirements . . .

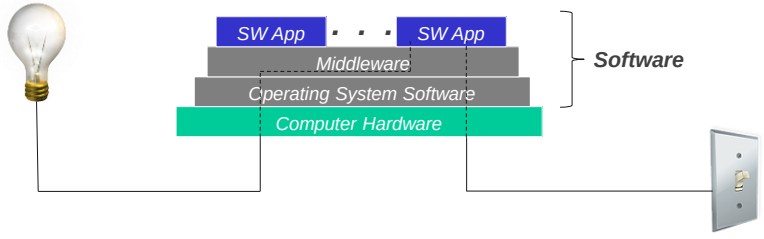


Background and Need (Cont'd)

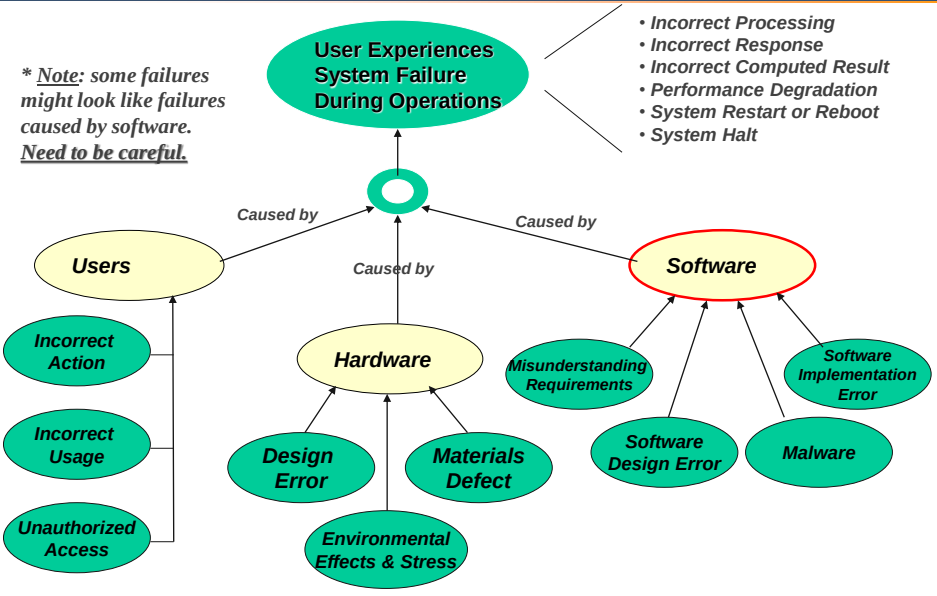
-- OK . . . So What About Software Safety Now?

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

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

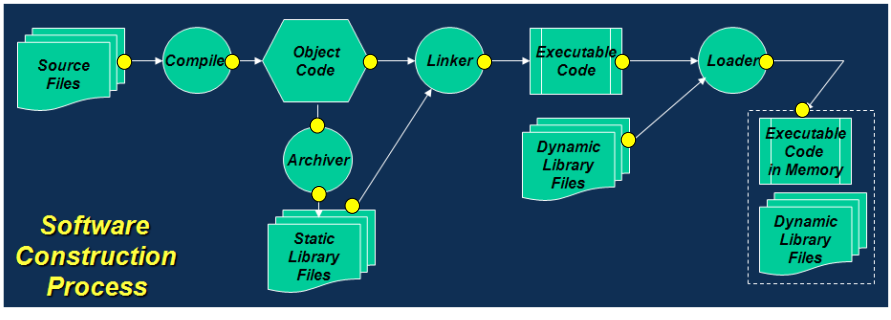


Background and Need (Cont'd)
-- Software Failure Causes



Background and Need (Cont'd)
-- Sources of Errors in Software Process

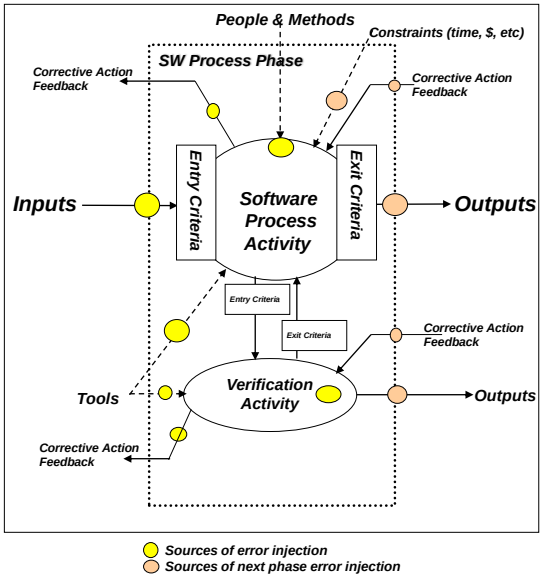
- **Causes of Software failures**
  - *Latent defects in the source code, library files*
  - *Latent defects in tools affecting code construction*
  - *Environmental conditions operational software is not programmed to handle*



Background and Need (Cont'd)
-- Software Behavior, Hazards, and Observations

- Software contributors to hazards may include defects, errors, or omissions
  - *May lead to failure of system to operate 'correctly' which could lead to a hazardous condition*
- Correctly implementing requirements that are unsafe will not prevent mishaps
- Many requirements have nothing to do with hazardous behavior or mishaps
- Incorrect software behavior may not lead to hazards or mishaps
- Correct software behavior may lead to hazards or mishaps

Background and Need (Cont'd)
-- Sources of Errors in Software Process

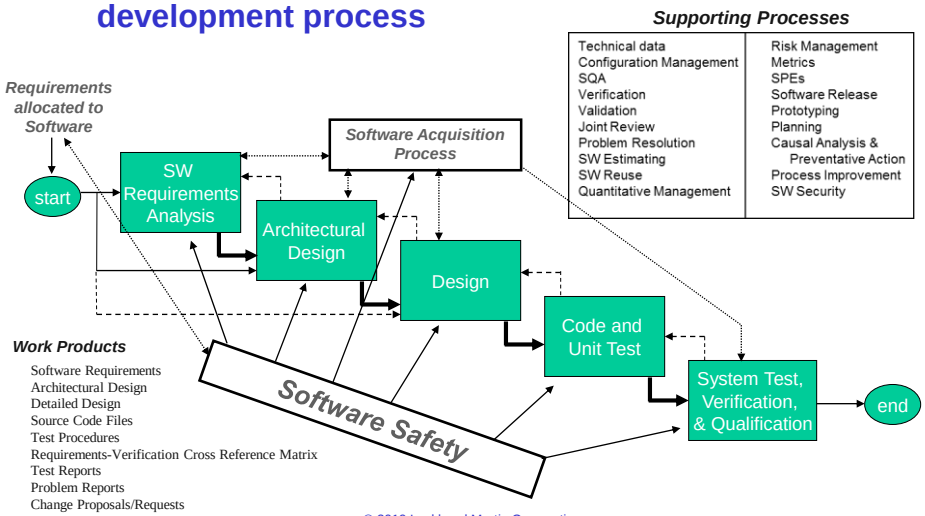


- Software Safety is not only about reducing error rates in safety-critical software (based on SCC)
- Software Safety is also about reducing the risk of software causing or inducing certain hazards that when realized, could lead to a system mishap, accident

Software Safety Process

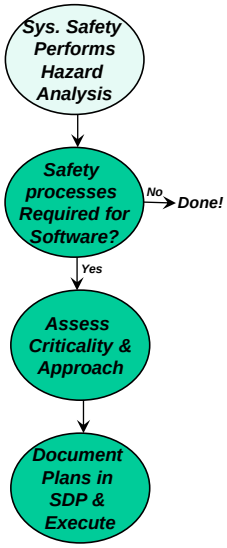
Software Safety Process
-- Software Process

- **Software Safety is integrated into the entire software development process**



Software Safety Process
-- General Approach

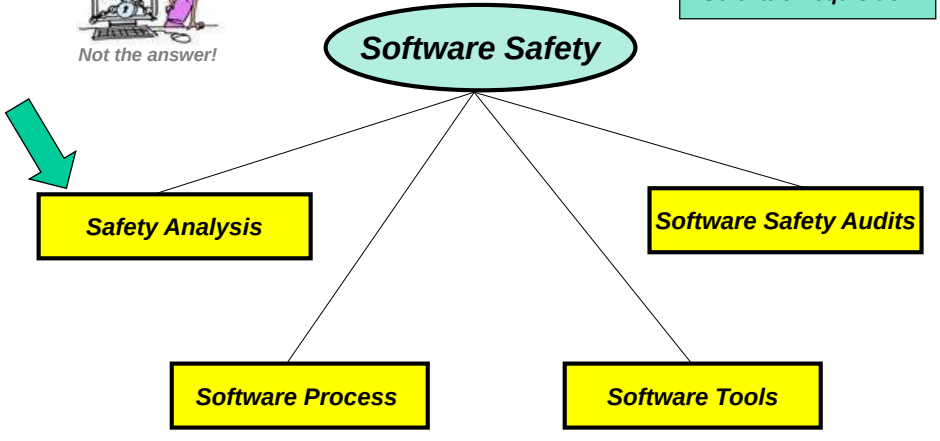
- **General Approach to safety in software . . .**
  - ***Participate in System Safety Analysis Activities***
    - Hazard analysis and other system safety team sponsored analyses
  - ***Adjust software process Level of Rigor (LOR) activities and/or software product design activities based on "Levels 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**



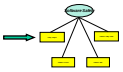
Software Safety Process


Not the answer!

- *Software Development*
- *Software Acquisition*



Software Safety Process
-- Safety Analysis

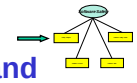


- **Software Engineering organization teams with Safety Engineering to understand hazards (i.e., risks and consequences) due to safety-critical functions failing**
  - ***Some critical functions may be monitored, controlled by software***
  - ***Safety Engineering Team responsible for hazard analysis***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

- **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

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

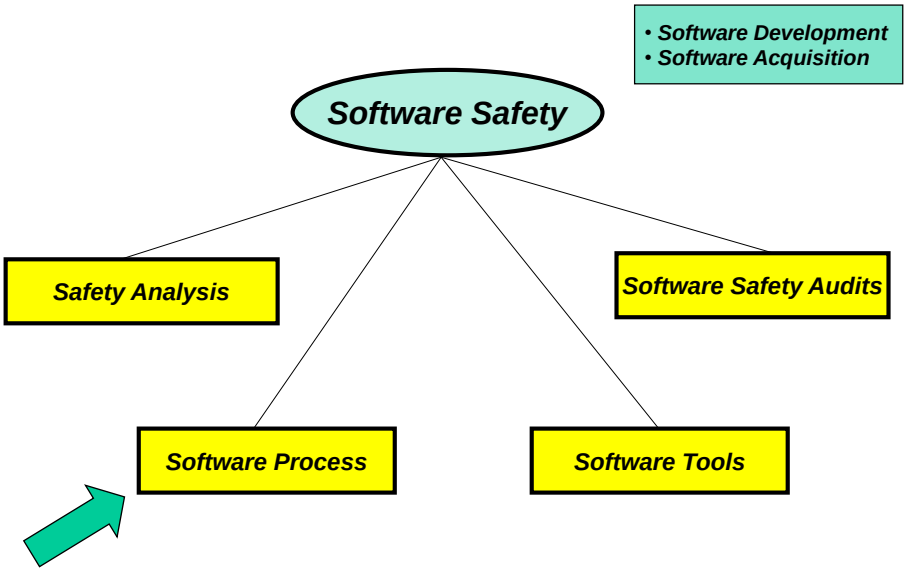
SFTA – Software Fault Tree Analysis
SFMEA – Software Failure Modes and Effects Analysis

Software Safety
-- 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)*

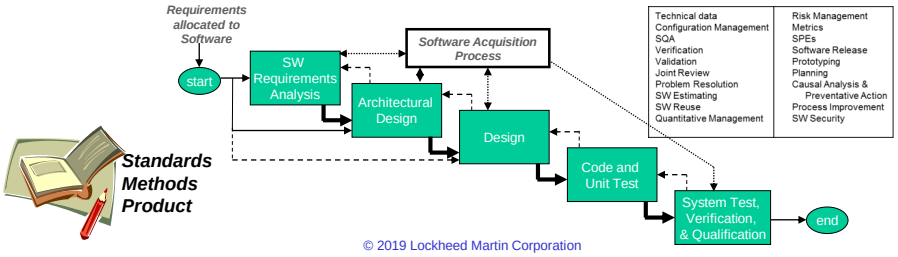
SWCI	Level of Rigor Tasks
SWCI 1	Program shall perform analysis of requirements, architecture, design, and code and conduct in-depth safety-specific testing.
SWCI 2	Program shall perform analysis of requirements, architecture, design, and conduct in-depth safety-specific testing.
SWCI 3	Program shall 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 Process



Software Safety Process
-- Software Process

- **Software Development Plan (SDP) captures software process, plans, and planning for software safety . . .**
  - ***Identification and Standards***
    - Identify safety-critical software components and standards in software process and product development activities
  - ***Software Methods***
    - Specify activities in software methods needed to address specifics of safety-critical software development
  - ***Software Product Assurance***
    - Specify product development activities needed to address quality management and metrics specifics of safety-critical software development



Standards
Methods
Product

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

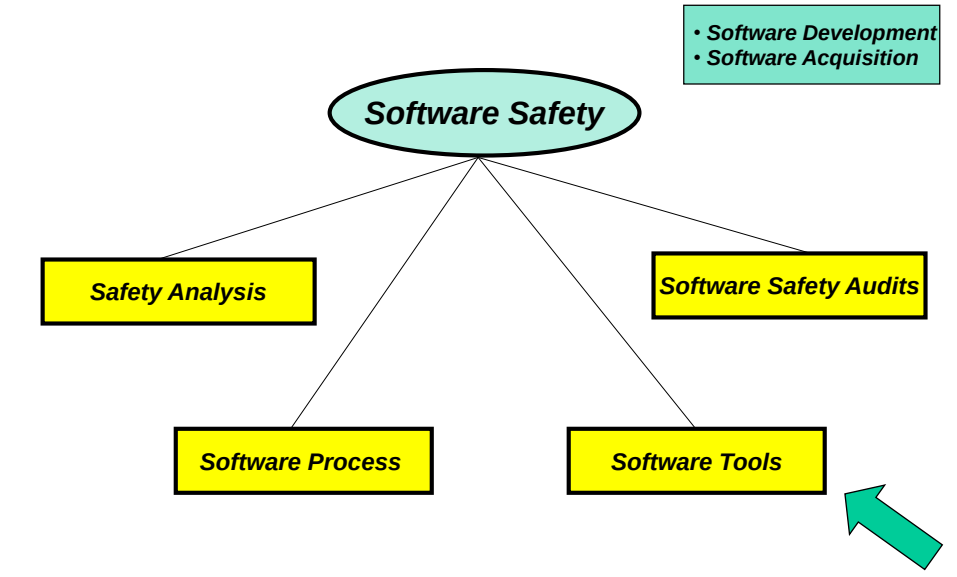
Software Safety Process
-- Software Process

- **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

Software Safety Process
-- Software Process

- **Software Product Assurance**
  - ***Mark requirements, design, code, and tests of safety-critical software***
  - ***Analysis and handling of dead code, deactivated code***
  - ***Verification of source in accordance with coding standards – automate checking, where practical***
    - Non-compliant software should be changed to be standard compliant or sufficient justification documented and reviewed by software mgmt. team
  - ***Specify functional, structural coverage and complexity metrics***
    - Specify thresholds where action is taken
  - ***Software quality growth, defect density, and defect resolution performance metrics***
  - ***Test for error propagation through software***
  - ***Test for failure modes involving software control or response***
  - ***Keep all software work products for safety-critical application current with changes to software***

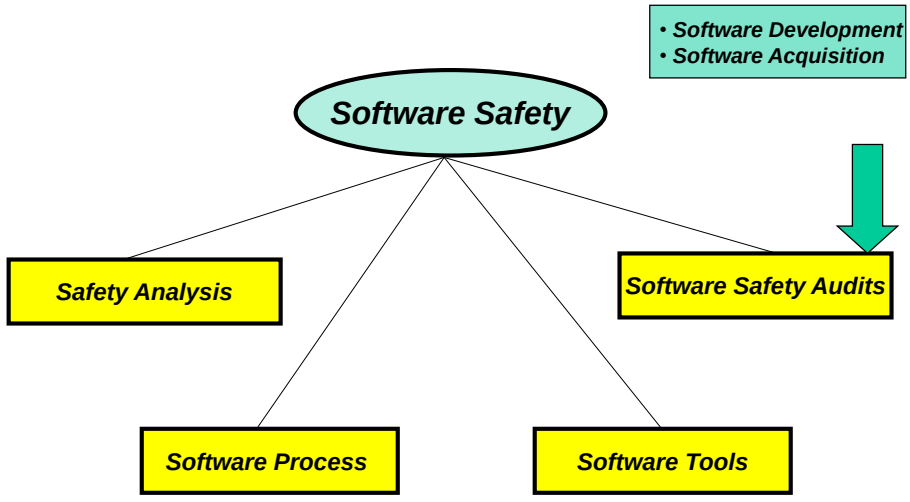
Software Safety Process



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

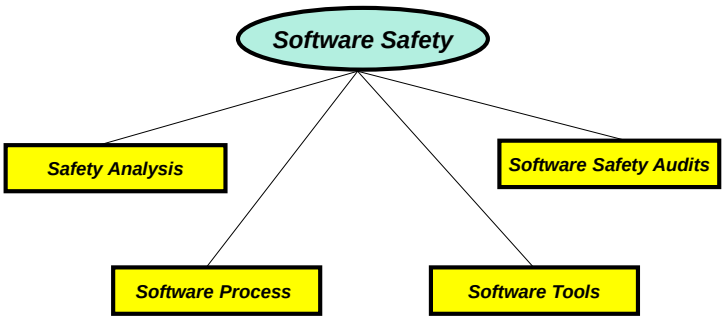


Software Safety Process
-- Software Safety Audits

- **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*"

Software Safety Process
-- Software Safety Audits

- *Software Development*
- *Software Acquisition*



Software Safety Process
-- Software Development

- **Software Development**
  - ***Participate in Systems Safety Analyses and reviews***
    - Identifies need for safety in software
    - Identifies what portions of software are of safety interest
  - ***Document approach to safety in Software Development Plan***
  - ***Conduct coordination review of SDP with safety group***
  - ***Assign "software safety engineer" role to software team member (software team safety advocate)***
  - ***Verify engineers developing safety-critical software are trained prior to developing safety-critical software, including program tools and metrics***
  - ***Include costs for development of safety-critical software in software cost estimates***

Software Safety Process
-- Software Acquisition

- **Software Acquisition**
  - ***Participate in System Safety Analyses and Reviews***
    - Identifies need for safety in software
    - Identifies what portions of software are of safety interest
  - ***Document approach to safety in Software Acquisition Management Planning***
    - Provide coordination review with safety group
  - ***Ensure Subcontractor's SDP accounts for how development of safety-critical software will be managed***
  - ***During reviews of subcontractor documentation . . .***
    - Ensure subcontractor's plans and planning for safety-critical software is based on criticality of software components and contract flowed requirements
      - *Hazard analyses, LOR*
  - ***Review subcontractor data products to . . .***
    - Ensure production and control of required SC work products (i.e., evidence for audit)
  - ***Include costs for development of safety-critical software in software cost estimates***
  - ***Support software safety audits***

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**

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

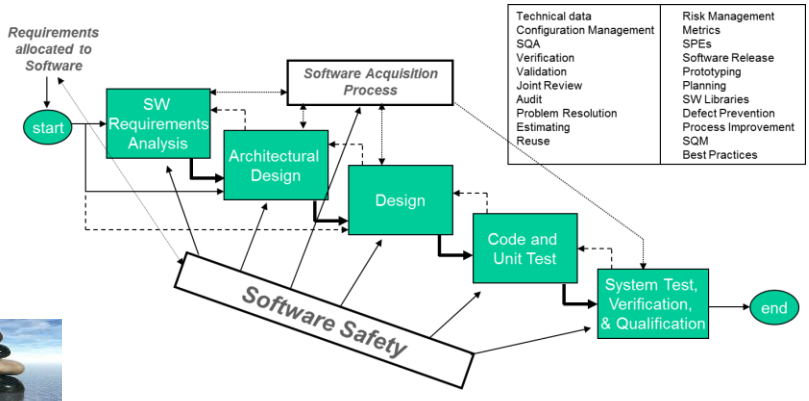
- **For highest levels of software assurance, may also require . . .**
  - ***Independence in verification activities***
  - ***Testing of every decision structure, every condition shown to take all possible outcomes at least once and each condition shown to affect outcome independently (MC/DC)***
  - ***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***

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***



Tailoring Guidance Example

Software Safety Process

Tailoring Guidelines

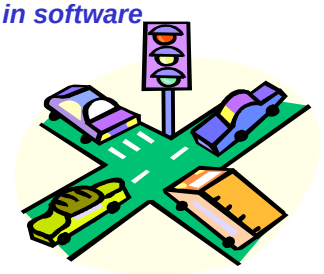
Reqt ID	Software Safety Practice Requirement	SWCI 1 Safety Critical	SWCI 2 Safety Significant	SWCI 3 Safety Related **
1	Identify the program safety levels of software with safety impact.	X	X	X
2	Identify and/or reference the software components associated with each program safety level.	X	X	X
3	Verify 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***

Exercise

Exercise

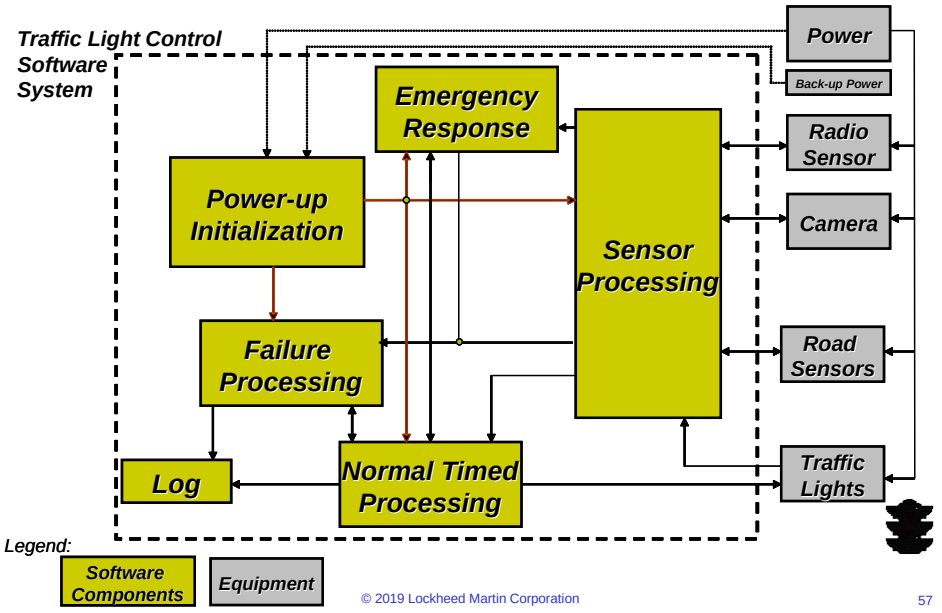
- **Real-world problem to understand application of software safety**
  - ***4-way Traffic Light at intersection of high-speed highways***
- **Exercise is to examine design of traffic light system, determine if software is safety-critical, and if so . . .**
  - ***Identify the levels of criticality and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



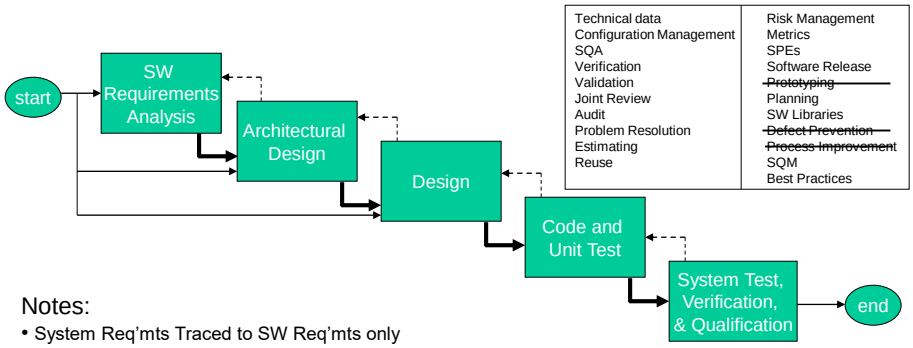
Exercise
-- Requirements (Example)

- **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. . . .***

Exercise
-- System/Software Functional Block Diagram (Example)



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

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 . . . .*

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

-- 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

-- 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

-- 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

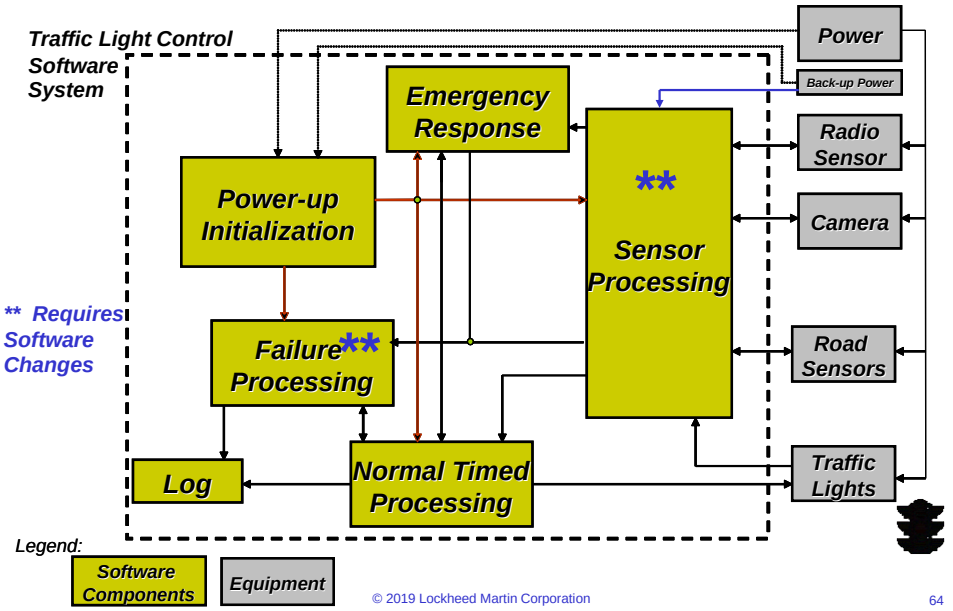
SWCI	Level of Rigor Tasks
SWCI 1	Program shall perform analysis of requirements, architecture, design, and code and conduct in-depth safety-specific testing.
SWCI 2	Program shall perform analysis of requirements, architecture, design, and conduct in-depth safety-specific testing.
SWCI 3	Program shall 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**

Exercise

-- System/Software Functional Block Diagram (Example)



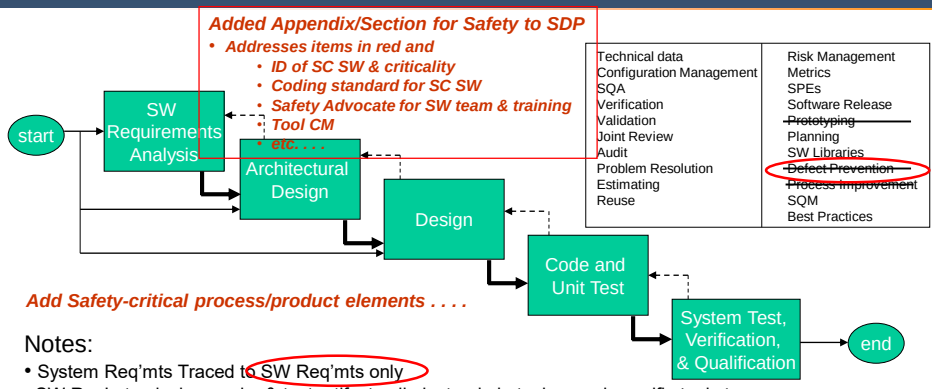
Course Exercise
-- Software Safety Process Tailoring

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

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

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

-- 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			

Exercise

Now it's your turn

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			

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.)

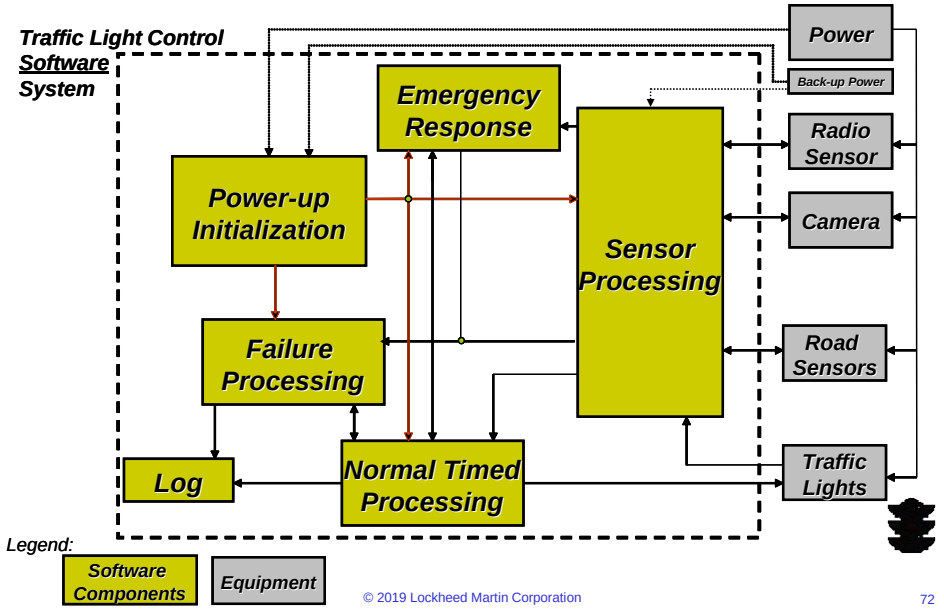
Controls:

Effects after Controls:

Remarks:

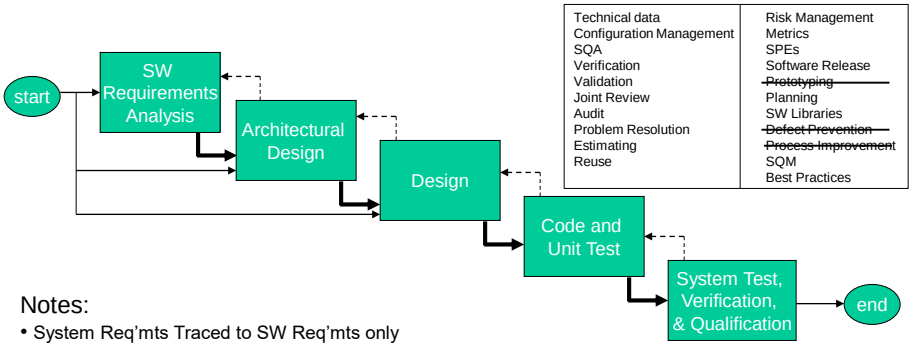
Hazard Closure Evidence:

Exercise
-- System/Software Functional Block Diagram



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

-- 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

-- 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)**

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			

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.

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

-- Exercise Hazards for Group Work

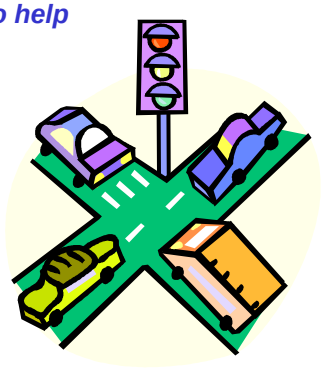
- Red/(green) lamp burns out on the N-S bound lane leading to no stop/(go) indication for on-coming traffic that did not see the previous traffic light transition.
- [Barn swallows build a nest on the traffic light fixture (unnoticed?).] The RF sensor circuit [is compromised and] fails to engage all-stop 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.

Course Exercise

- 20 - 30 Minutes

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 . . . .**



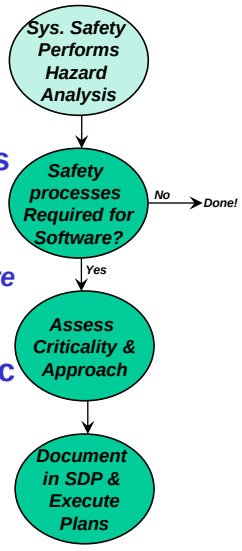
Summary

Summary

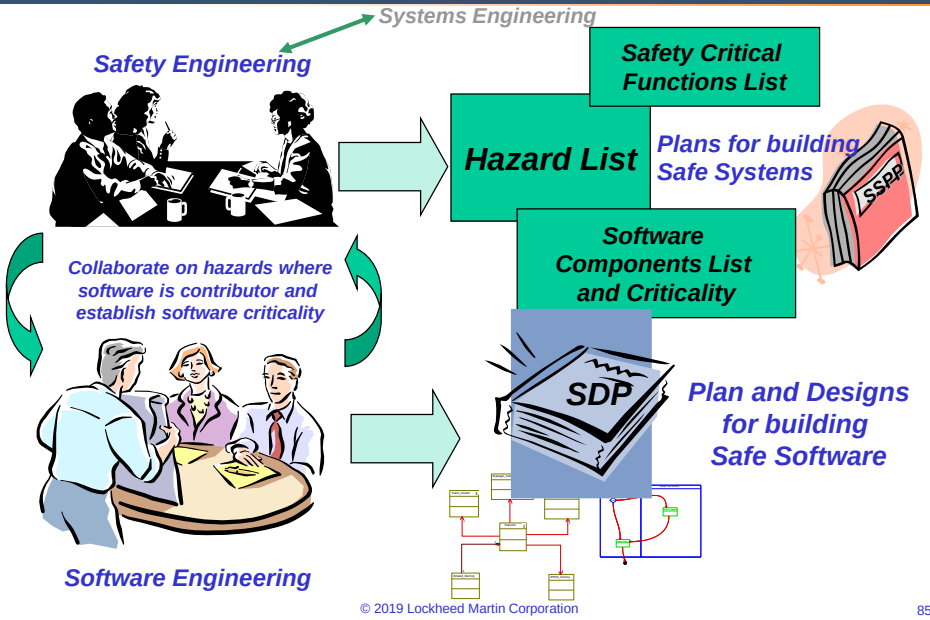
- **Safe Software ≠**
  - *Lower software defect rates*
  - *Reliable Software*
  - *Secure Software*
- **Safety is a systems attribute**
  - *Software Engineering and software are contributors to safe systems and safe operations*
- **Safety Engineering conducts hazard analysis on program**
  - *Software 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*

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)*



Summary (Cont'd)



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 . . . .*

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

Glossary

- *Certification* – legal recognition that a product, service, organization, or person complies with requirements. The activity involves technically checking the product, service, organization, or person and the formal recognition of compliance with the requirement by issue of a certificate or license in compliance with governing law.
- *Condition/Decision Coverage* – every point of entry and exit of a program has been invoked at least once and every condition in a decision has taken all possible outcomes at least once and every decision has taken on all possible outcomes at least once.
- *Designated Engineering Representative (DER)* -- any properly qualified private person or employee to which the FAA has delegated responsibility for any work, business, or function with respect to the examination, inspection, and testing necessary to the issuance of certificates in accordance with FAA standards.
- *Deactivated Code* – executable code that is not intended by design to be executed or used in specific configurations of a target system.
- *Dead Code* – executable code that as a result of a design error cannot be executed or used and is not traceable to a requirement
- *Decision Coverage* – every point of entry and exit of a program has been invoked at least once during testing and every decision has taken on all possible outcomes at least once.
- *Error* – a mistake in the requirements, design, or code of the software
- *Failure* – inability of the software to perform its intended function within specified limits or constraints.
- *Fault* – a manifestation of an error. A fault may cause a failure.
- *Fault Tolerance* – the capability of a system to provide continued correct operation even in the presence of a limited set of equipment or software faults
- *Independence* – different teams with limited interactions developed portions or aspects of the software or software work products. A separation of responsibilities.
- *Modified Condition/Decision Coverage* -- a form of exhaustive testing where all of the following must be true at least once: (1) Each decision tries every possible outcome, (2) Each condition in a decision takes on every possible outcome, (3) Each entry and exit point to/from the program is invoked, and (4) Each condition in a decision is shown to independently affect the outcome of the decision. Independence of a condition is shown by proving that only one condition changes at a time.
- *Safety-Critical Function* -- 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
- *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
- *Software Safety Assessment* – the activities that demonstrate compliance with airworthiness requirements. These may include functional hazard assessment, preliminary safety assessment, and system safety assessment, the rigor of which is related to the criticality of the system .
- *User-Modifiable Software* – software intended to be modified by an operator without review of a certifying authority if this modification is within the design constraints of the software established prior to the certification.

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 . . .

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

Lockheed Martin Aeronautics Company

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

Lockheed Martin Aeronautics Company