LOCKHEED MARTIN

Aeronautics

Software Safety
-- Process Overview and Application

Dr. Michael F. Siok, PE, ESEP
Lockheed Martin Aeronautics Company
P.O. Box 748, MZ 5924
Fort Worth, TX 76101
Tel: (817) 777-4234
FAX: (817) 762-9428
Email: Mike.F.Siok@lmco.com

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 recent 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 Atlanta Sky-high water bills**
  - ***. . . Software in 450 water meters miscalculated usage and charged homeowners more than they should have . . . ."***

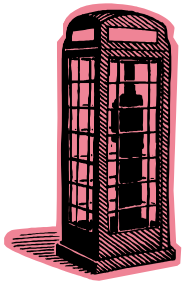
Software Failures Affect Society

… a few recent 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 . . . .



Course Topics

-- Software Safety

- **Background and Need**
- **Software Safety Process**
- **Exercise**
- **Summary and Close**

Background and Need

Background and Need

- **Software Safety can only be considered in context of an *<u>Operational System</u>***
  - ***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)

- 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 882D, ARP-4761*
  - *Software not excluded from safe systems operation*

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

Background and Need (Cont'd)

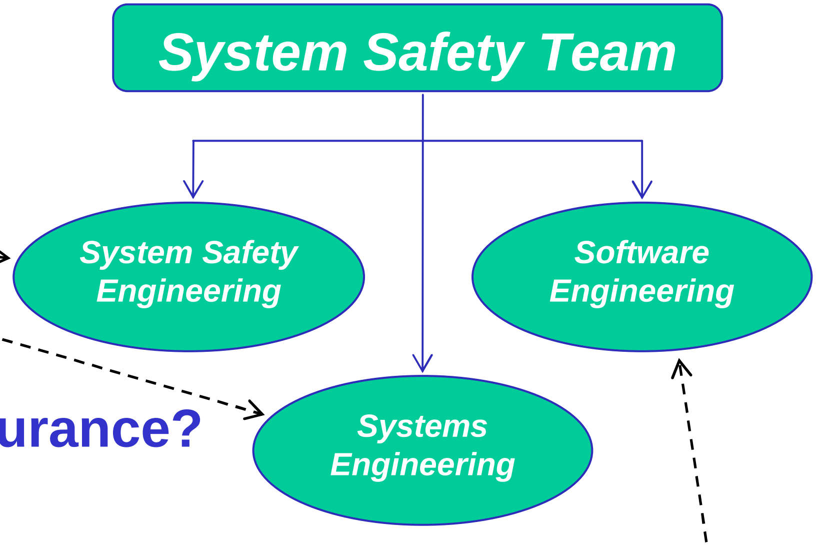
- **Definitions:**
  - *Safety-Critical Software*
    - A software unit, component, object, or software system whose ***proper recognition, control, performance, or fault tolerance is essential to the safe operation*** and support of the system in which it executes.
  - *Safety-Critical Functions*
    - 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.

How Do We ID Critical Software Processing?

- <u>DEF: Software Safety</u> -- application of disciplined system safety engineering, systems engineering, and software engineering to ensure 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 Software Safety Assurance?
  - *SW Architecture & Design*
  - *SW Processes & Methods*
  - *SW Tooling*



System Safety Team

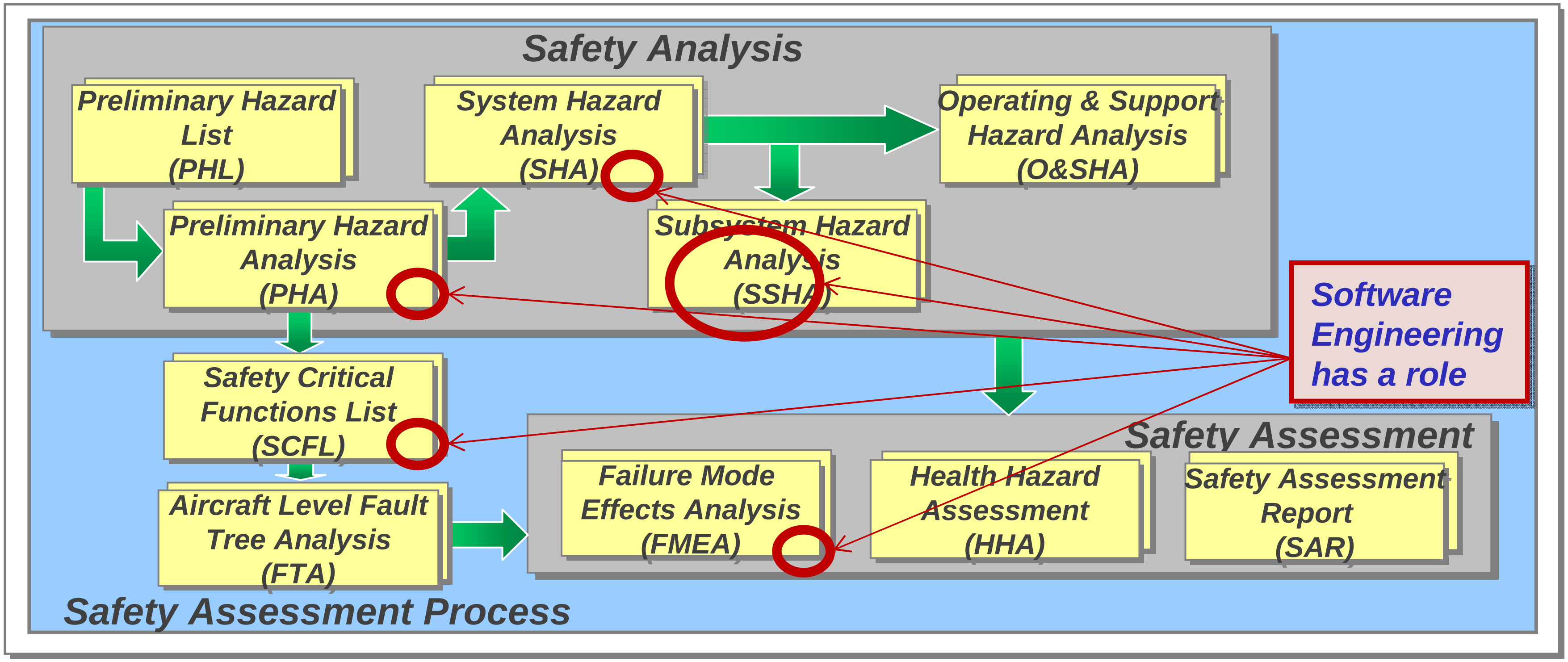
System Safety Engineering

Software Engineering

Systems Engineering

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.*



Safety Analysis
Preliminary Hazard List (PHL)
System Hazard Analysis (SHA)
Operating & Support Hazard Analysis (O&SHA)
Preliminary Hazard Analysis (PHA)
Subsystem Hazard Analysis (SSHA)
Software Engineering has a role
Safety Critical Functions List (SCFL)
Aircraft Level Fault Tree Analysis (FTA)
Safety Assessment
Failure Mode Effects Analysis (FMEA)
Health Hazard Assessment (HHA)
Safety Assessment Report (SAR)
Safety Assessment Process

Background and Need (Cont'd)

- **Goal of <u>Software</u> System Safety Program**
  - ***Integrate seamlessly with <u>System</u> 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 882D Mishap Severity Categories

- ***Catastrophic*** – could result in death, permanent total disability, loss exceeding $1M, severe environmental damage violating law or regulation
- ***Critical*** – could result in permanent partial disability, injuries or illness affecting at least 3 people, loss >$200K but <$1M, or reversible damage to environment but violating law or regulation
- ***Marginal*** – could result in injury or illness resulting in loss of 1 or more work days, loss >$10K but <$200K, mitigatable environmental damage without violation of law or regulation where restoration activities can be accomplished
- ***Negligible*** – Could result in injury or illness not resulting in lost workdays, loss >$2K but <$10K, minimal environmental damage not violating law or regulations

MIL-STD-882D Hazard Severity Levels	*UK DEF-STAN-00-55 Software Safety Integrity Levels* *	*RTCA/DO-178B Software Levels*	*Standard Model Software Criticality*
I Catastrophic	*SIL 4*	*A Catastrophic*	*Safety Critical*
II Critical	*SIL 3*	*B Hazardous*	*Safety Significant*
III Marginal	*SIL 2*	*C Major*	*Safety Related*
IV Negligible	*SIL 1*	*D Minor*	*Minor Safety Impact*
---	*SIL 1*	*E No Effect*	*No Safety Impact*

Levels of Software Safety Criticality

Background and Need (Cont'd)

-- MIL-STD 882D Mishap Severity Categories

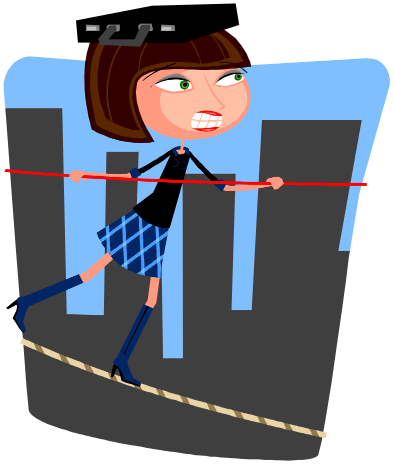
Hazard Severity Category		Hazard Probability				
		FREQUENT A	PROBABLE B	OCCASIONAL C	REMOTE D	IMPROBABLE E
CATASTROPHIC - Safety Critical event resulting in: death, aircraft loss or damage beyond economical repair; or severe environmental damage	I	HRI 1	HRI 2	HRI 4	HRI 8	HRI 11
CRITICAL - Safety Critical event resulting in: Severe injury or occupational illness to any personnel that results in a permanent partial disability; aircraft or property damage > $1,000,000; a condition that requires immediate action to prevent the above (including Cat I) or major environmental damage.	II	HRI 3	HRI 5	HRI 6	HRI 10	HRI 15
MARGINAL - Minor injury or occupational illness; aircraft or property damage > $ 20,000; an In-flight failure requiring termination of flight for safety reasons or correctable environmental damage.	III	HRI 7	HRI 9	HRI 12	HRI 14	HRI 17
NEGLIGIBLE - Less than minor injury or damage or minimal environmental damage. Mission can be continued with minimum risk.	IV	HRI 13	HRI 16	HRI 18	HRI 19	HRI 20

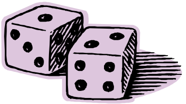
	MIL-STD-882D Hazard Severity Levels	*UK DEF-STAN-00-55 Software Safety Integrity Levels **	*RTCA/DO-178B Software Levels*	*Standard Model Software Criticality*
HRI 1 – 3	*I Catastrophic*	*SIL 4*	*A Catastrophic*	*Safety Critical*
HRI 4 – 7	*II Critical*	*SIL 3*	*B Hazardous*	*Safety Significant*
HRI 8 – 10	*III Marginal*	*SIL 2*	*C Major*	*Safety Related*
HRI 11 -- 20	*IV Negligible*	*SIL 1*	*D Minor*	*Minor Safety Impact*
	---	*SIL 1*	*E No Effect*	*No Safety Impact*

Background and Need (Cont'd)

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

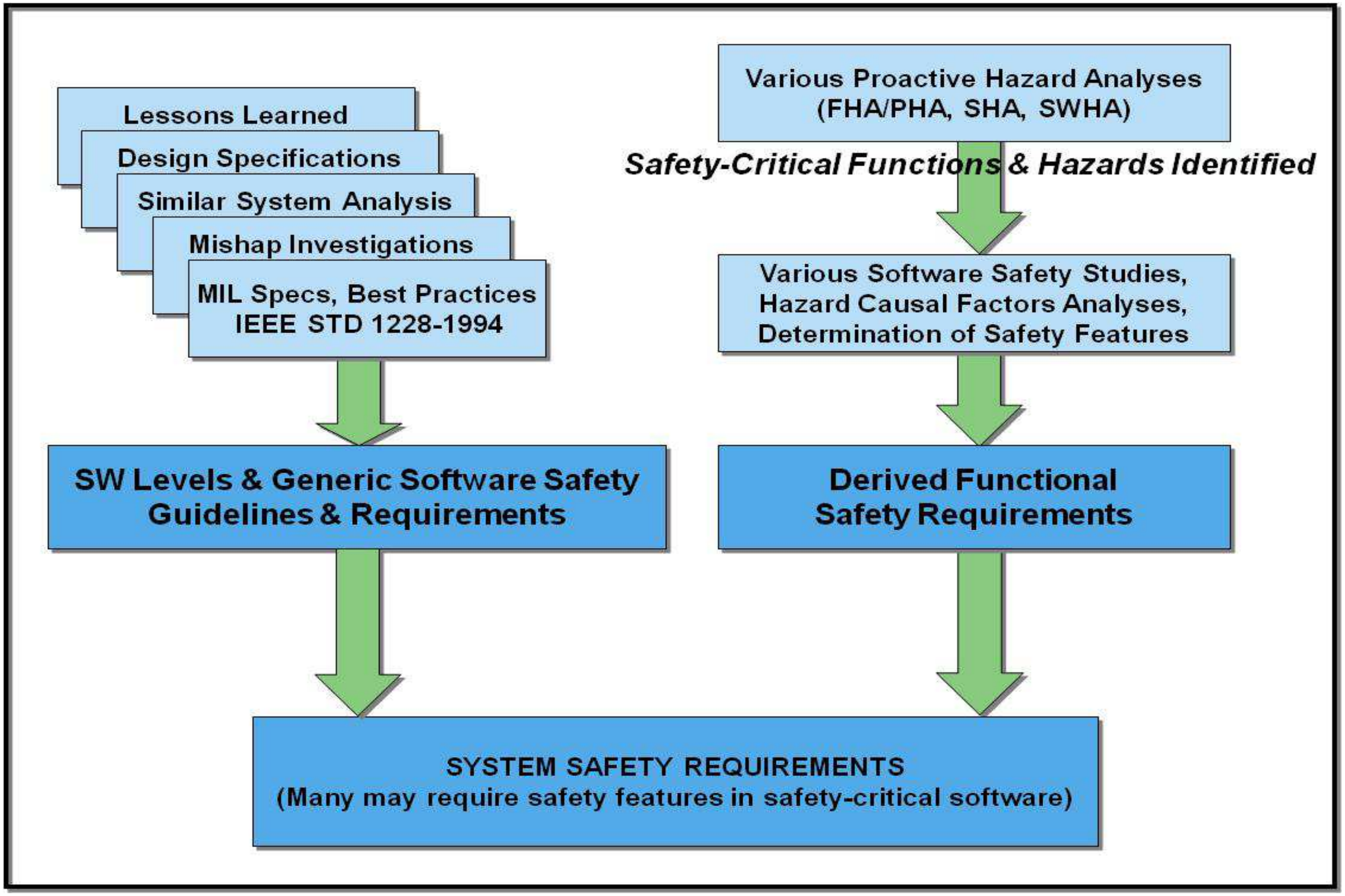
- **3 Assessment Areas for Safety Risk Consequence**
  - ***Person or people***
    - Death
    - Disability
    - Injury
    - Illness
  - ***Financial Loss***
    - $ millions or more
    - Negligible
  - ***Damage to Environment***
    - Reversible or Severe Damage
    - Break Regulations or Laws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





Background and Need (Cont'd)

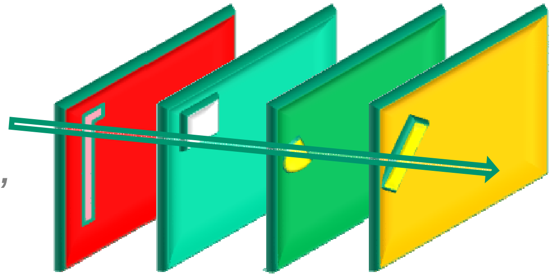
-- From Hazards to Requirements . . .



Lessons Learned
Design Specifications
Similar System Analysis
Mishap Investigations
MIL Specs, Best Practices
IEEE STD 1228-1994
Various Proactive Hazard Analyses
(FHA/PHA, SHA, SWHA)
Safety-Critical Functions & Hazards Identified
Various Software Safety Studies,
Hazard Causal Factors Analyses,
Determination of Safety Features
SW Levels & Generic Software Safety
Guidelines & Requirements
Derived Functional
Safety Requirements
SYSTEM SAFETY REQUIREMENTS
(Many may require safety features in safety-critical software)

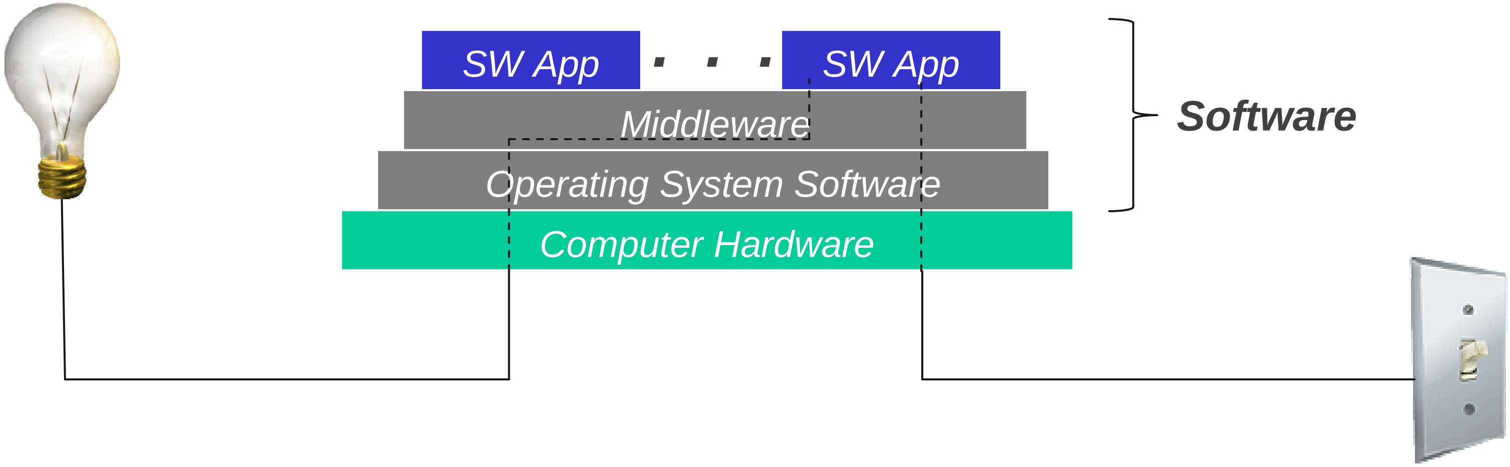
Background and Need (Cont'd)

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



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

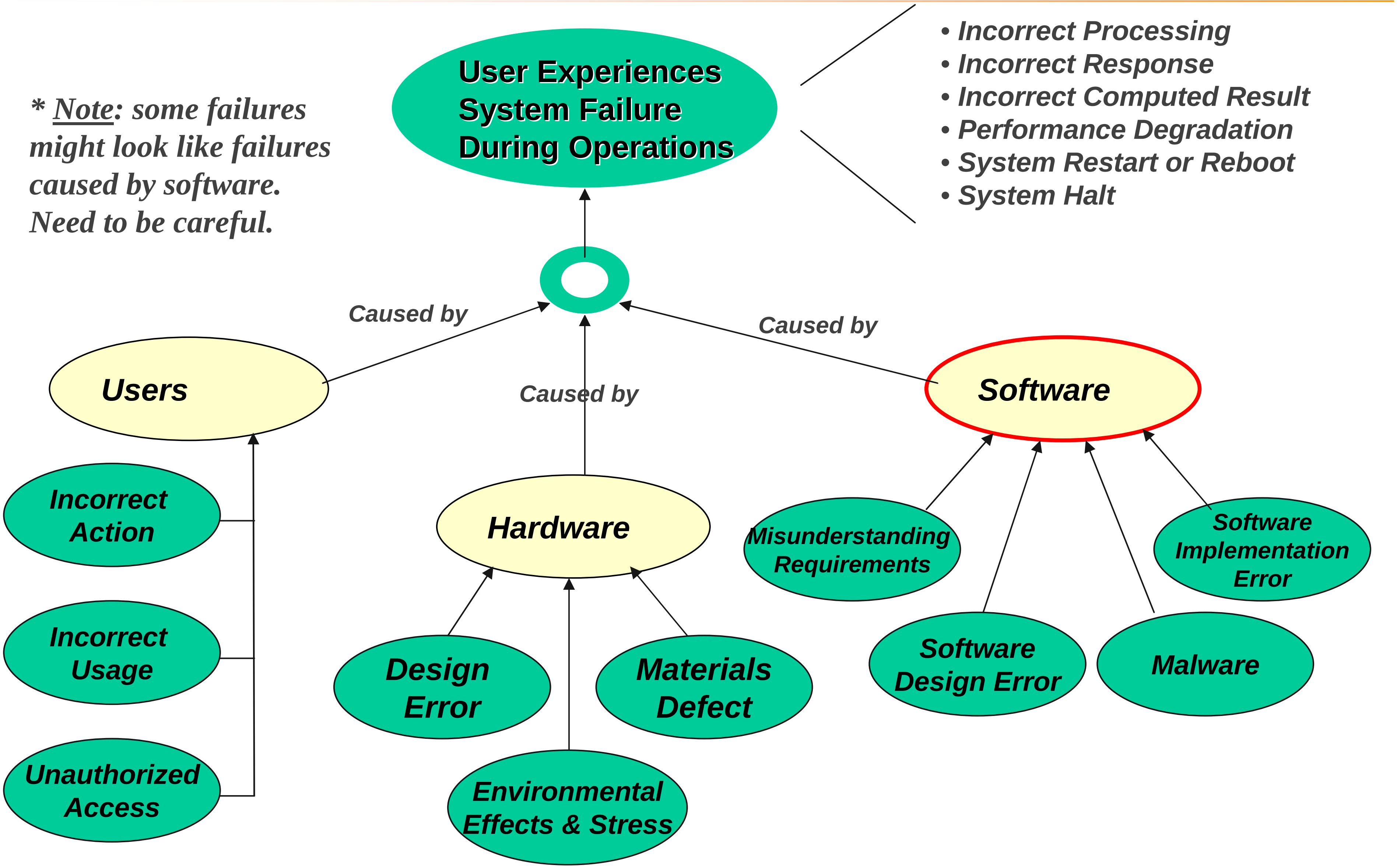
How can *Software* cause mishaps or accidents???



Background and Need (Cont'd)

-- Software Failure Causes

* *Note*: some failures might look like failures caused by software. Need to be careful.



User Experiences System Failure During Operations

- Incorrect Processing
- Incorrect Response
- Incorrect Computed Result
- Performance Degradation
- System Restart or Reboot
- System Halt

Caused by

Caused by

Caused by

Users

Hardware

Software

Incorrect Action

Incorrect Usage

Unauthorized Access

Design Error

Materials Defect

Environmental Effects & Stress

Misunderstanding Requirements

Software Implementation Error

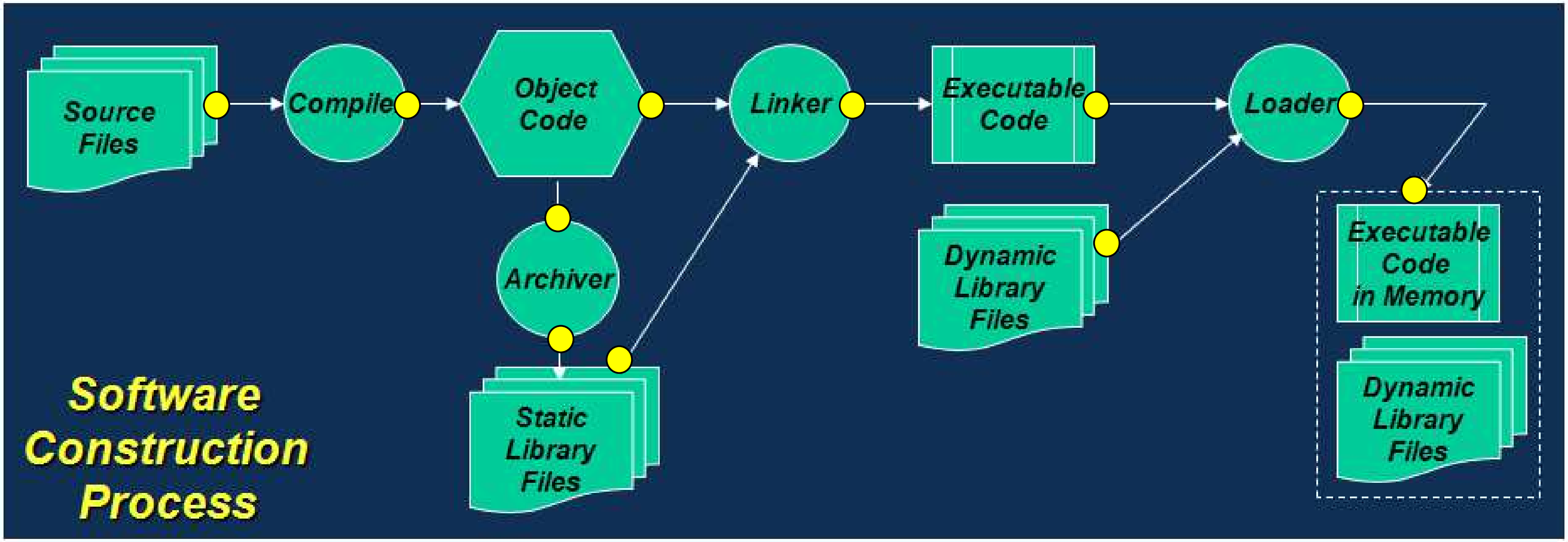
Software Design Error

Malware

Background and Need (Cont'd)

-- Sources of Errors in Software Process

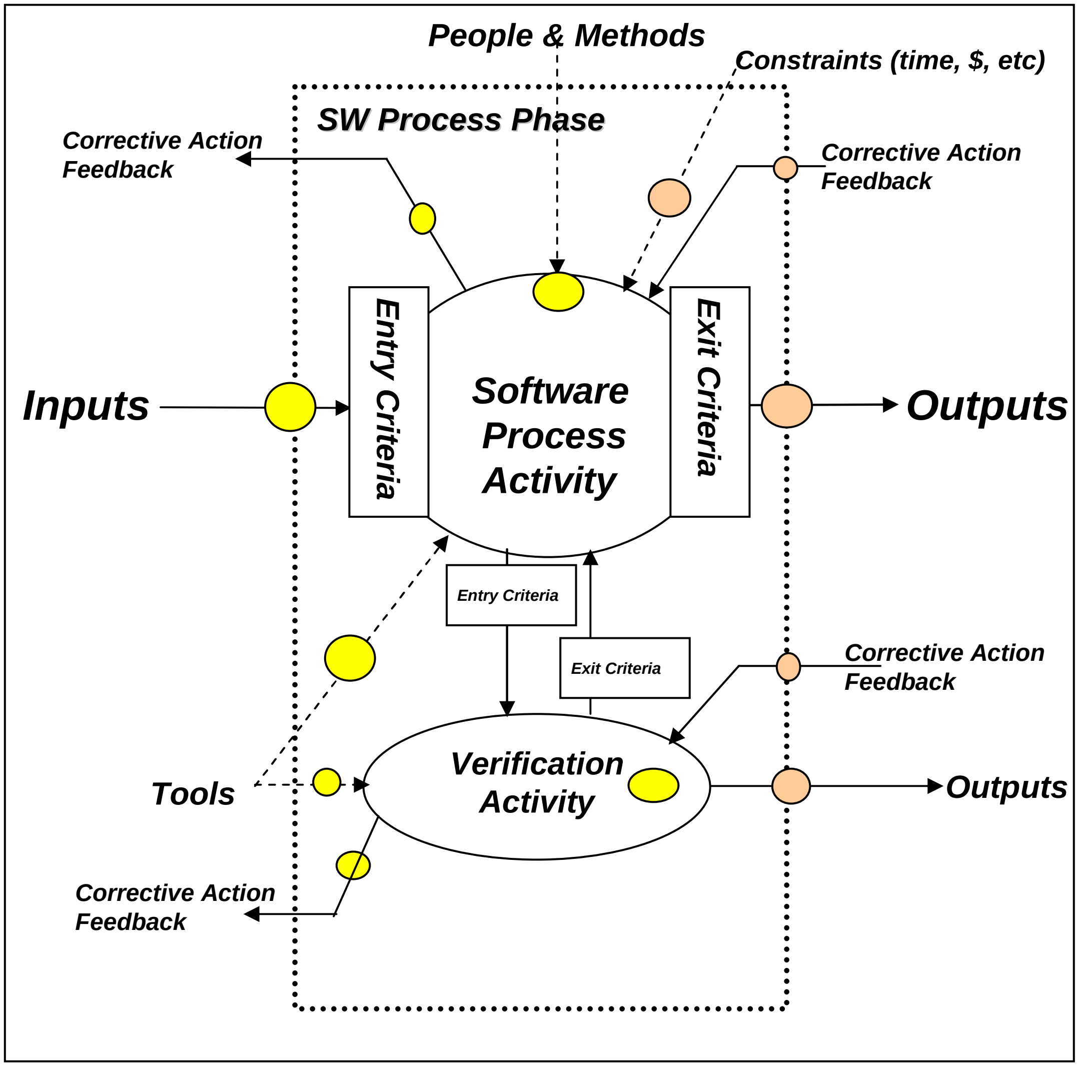
- Causes of Software failures
  - *Latent defects in the source code, library files*
  - *Latent defects in tools affecting code construction*
  - *Environmental conditions operational software is not programmed to handle*



● *Sources of error injection*

Background and Need (Cont'd)

-- Sources of Errors in Software Process


People & Methods
Constraints (time, $, etc)
SW Process Phase
Corrective Action Feedback
Corrective Action Feedback
Inputs
Entry Criteria
Software Process Activity
Exit Criteria
Outputs
Entry Criteria
Exit Criteria
Corrective Action Feedback
Verification Activity
Tools
Outputs
Corrective Action Feedback

🟡 *Sources of error injection*
🟠 *Sources of next phase error injection*

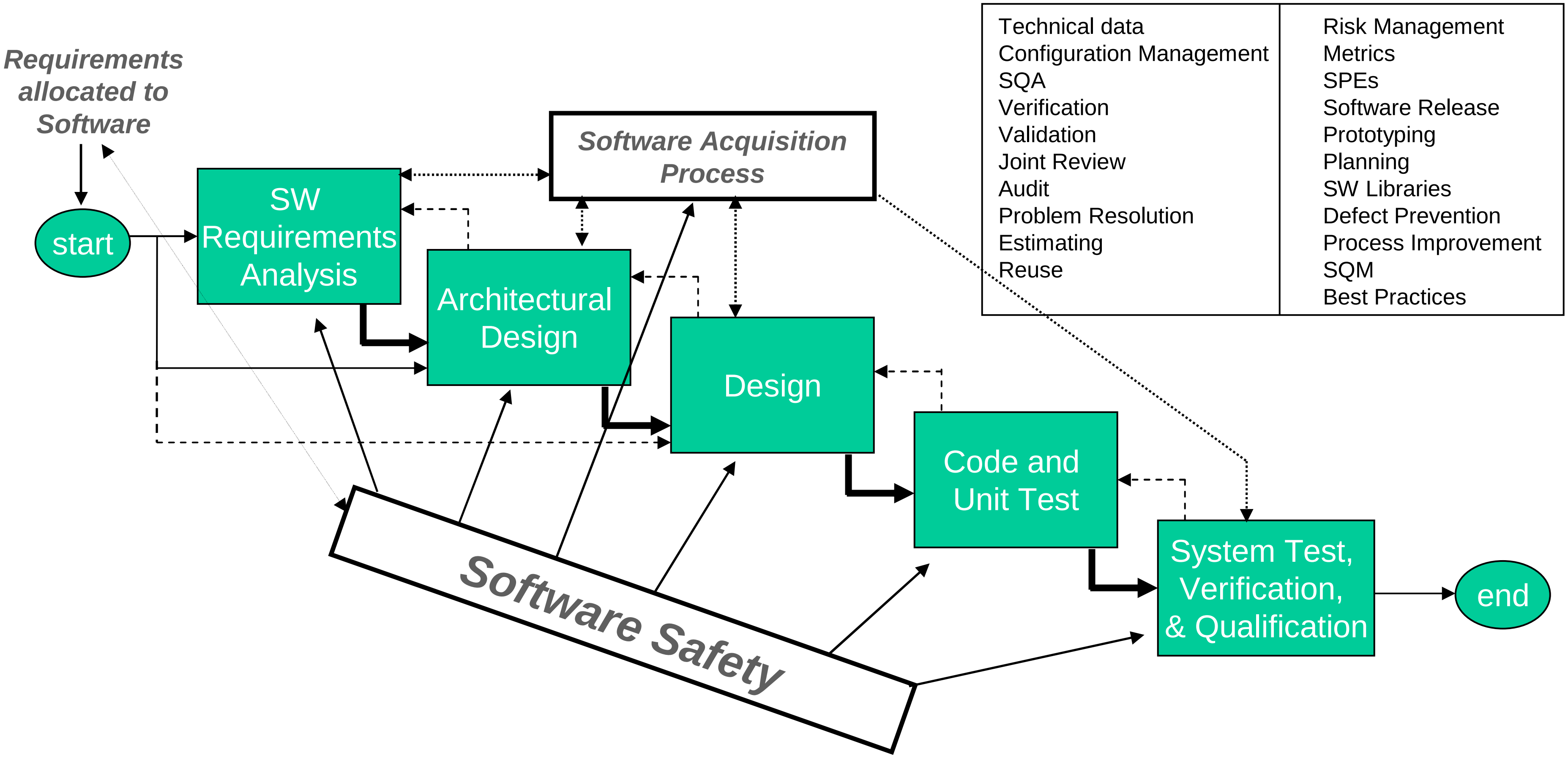
- Software Safety is <u>not only</u> about reducing error rates in software process or products
- Software Safety is about reducing the risk, in software intensive systems, of software causing or inducing certain hazards that when realized, could lead to a system mishap

Software Safety Process

Software Safety Process

-- Software Process

- Software Safety is integrated into the entire software development process



Requirements allocated to Software

start

SW Requirements Analysis

Architectural Design

Design

Code and Unit Test

System Test, Verification, & Qualification

end

Software Acquisition Process

Software Safety

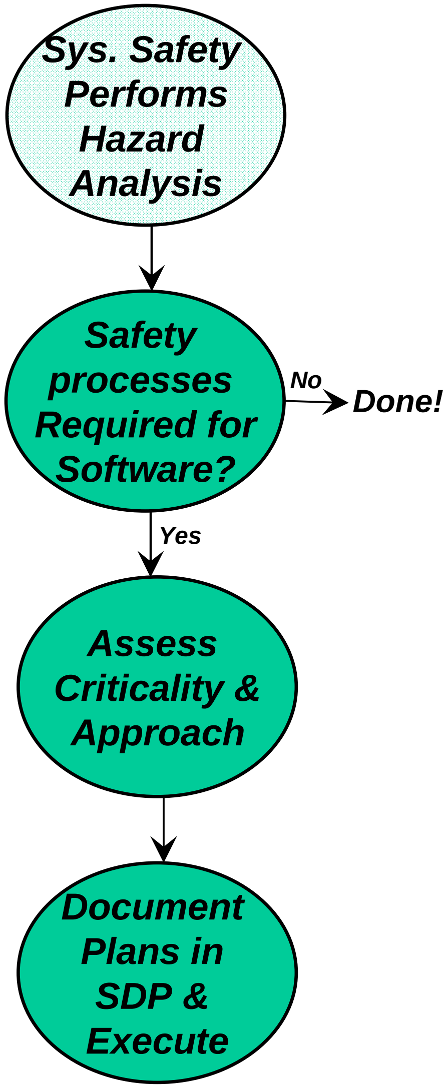
Technical data
Configuration Management
SQA
Verification
Validation
Joint Review
Audit
Problem Resolution
Estimating
Reuse

Risk Management
Metrics
SPEs
Software Release
Prototyping
Planning
SW Libraries
Defect Prevention
Process Improvement
SQM
Best Practices

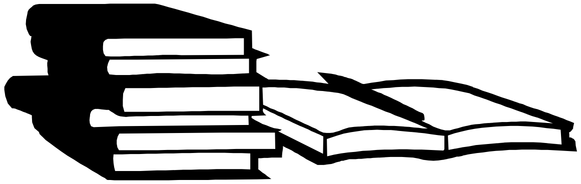
Software Safety Process

-- General Approach

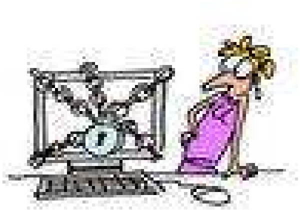
- General Approach to safety in software (at LM Aero) . . .
  - *Participate in System Safety Analysis Activities*
    - Hazard analysis and other system safety team sponsored analyses
  - *Adjust software process activities and/or software product design based on "Levels of Safety Criticality"*
   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


Sys. Safety Performs Hazard Analysis
Safety processes Required for Software?
No
Done!
Yes
Assess Criticality & Approach
Document Plans in SDP & Execute

- Software Development Plan (SDP) and/or Software Acquisition Management Plan (SAMP) documents project approach to safety in software

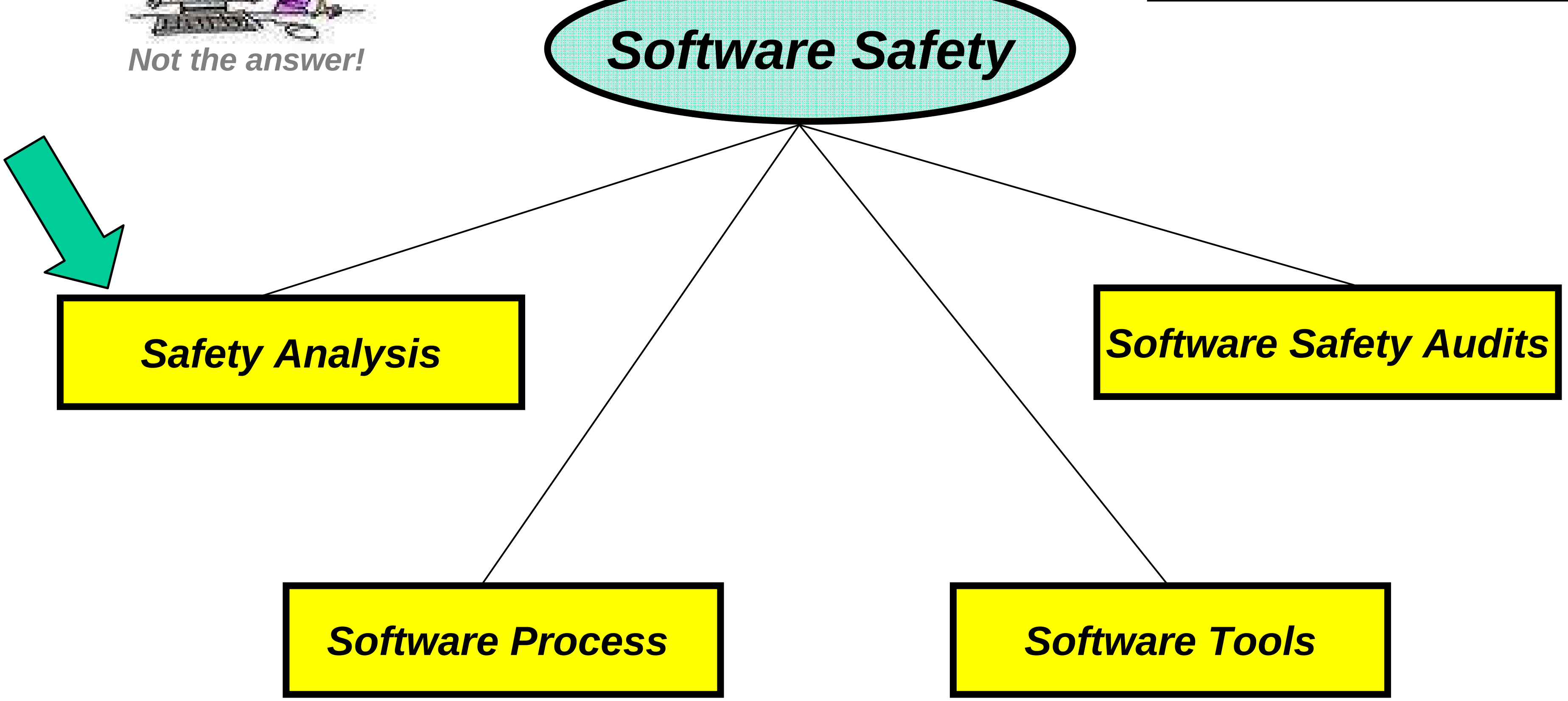


Software Safety Process



Not the answer!

- ***Software Development***
- ***Software Acquisition***



Software Safety

Safety Analysis

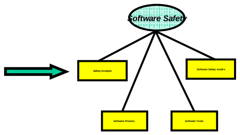
Software Safety Audits

Software Process

Software Tools

Software Safety Process

-- Safety Analysis



Software Safety

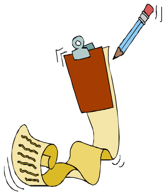
Safety Analysis

Software Safety Audits

Software Process

Software Tools

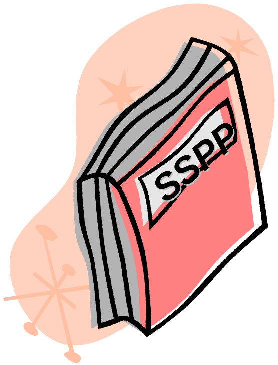
- Software Engineering organization teams with Safety Engineering to understand hazards (i.e., risks and consequences) and possible mishaps due to critical functions failing
  - *Some critical functions may be monitored, controlled by software*
  - *Safety Engineering Team responsible for hazard analysis*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



Hazards List

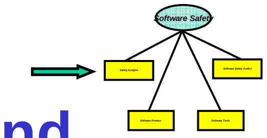
- Leads to list of safety-critical functions, level of criticality, and ID of software components that require special handling

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

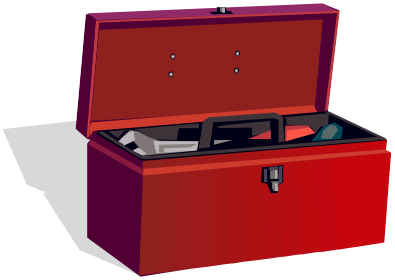


Software Safety Process

-- Safety Analysis (Cont'd)

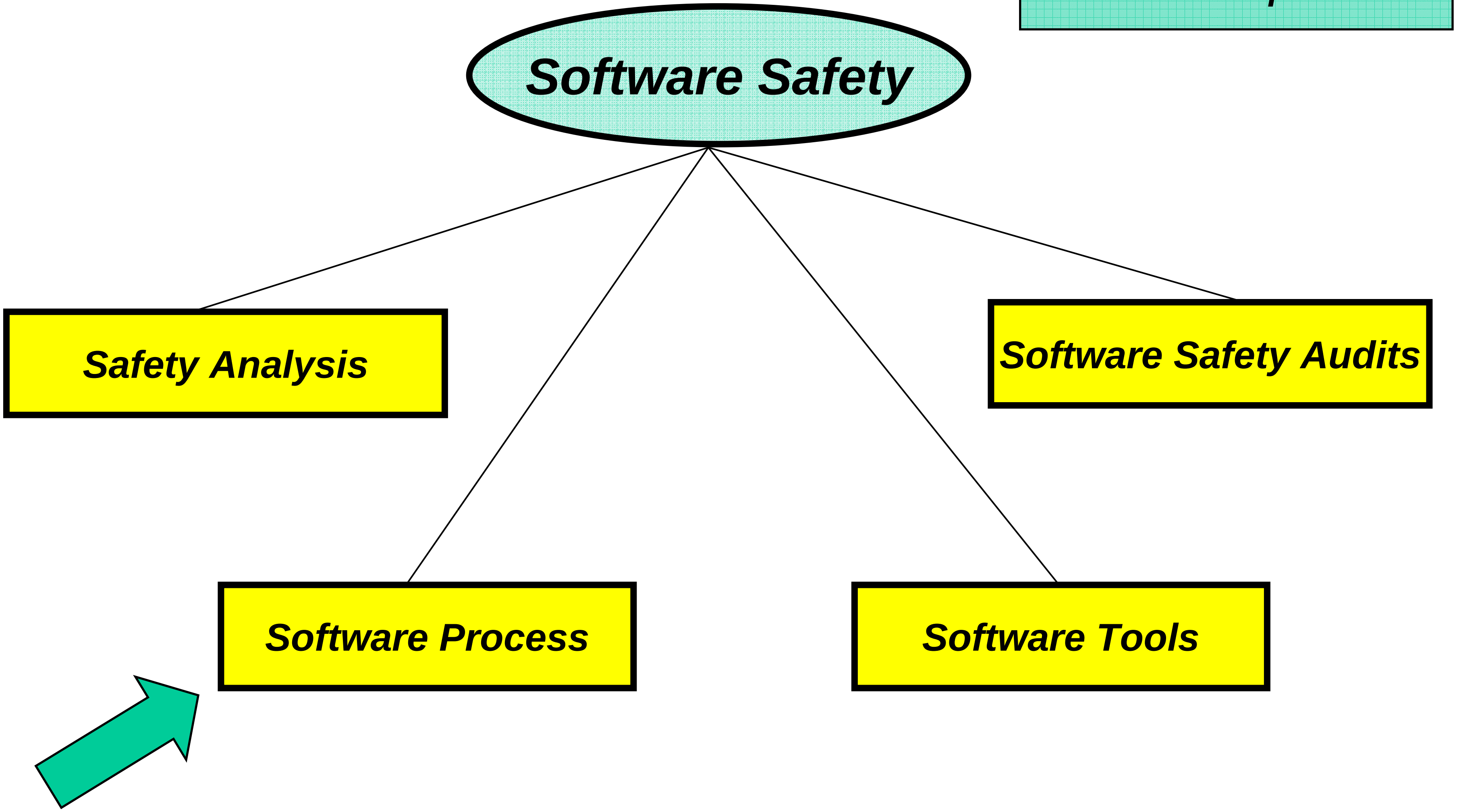


- **SW Engineering helps Safety Team identify appropriate risk reduction techniques to hazards and safety requirements through combination of . . .**
  - ***Software Analysis and Design Choices***
    - Safety-critical software identification
    - Safety interlocks, HW/SW Trades, partitioning, fault tolerance, etc.
    - Requirements, Design, and Coding Standardization
    - Safety Methods for software (SFTA, SFMEA, others)
  - ***Software Process***
    - 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



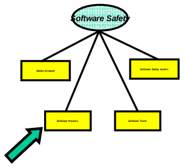
Software Safety Process

- Software Development
- Software Acquisition

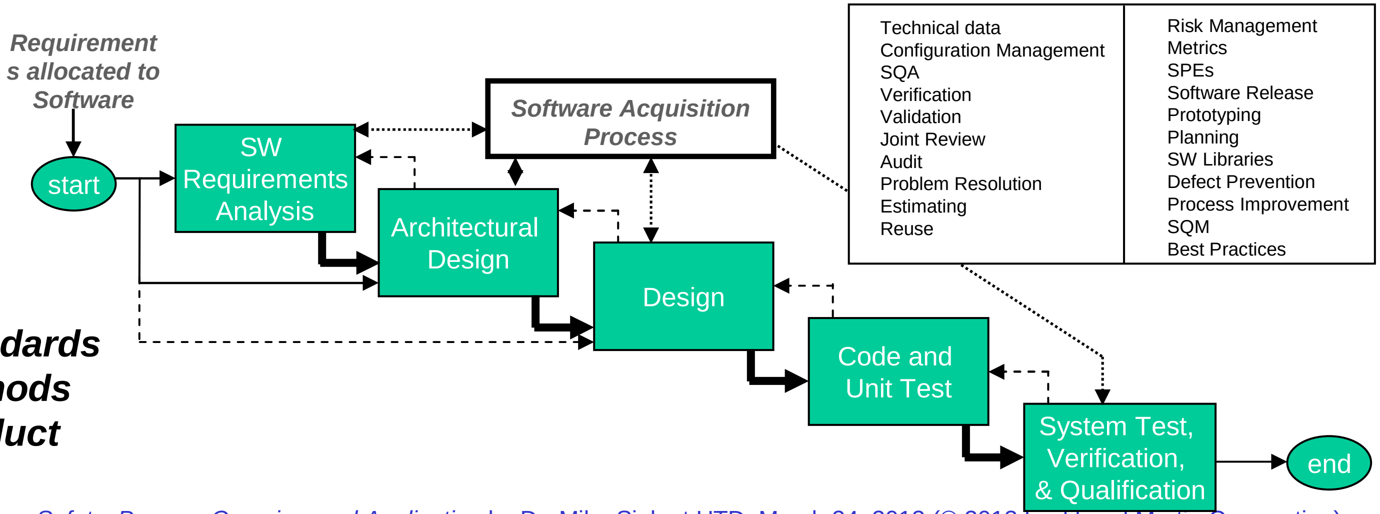


Software Safety Process

-- 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***
    - Identify activities and software methods needed to address specifics of safety-critical software development
  - ***Software Product Assurance***
    - Identify product development activities needed to address quality management and metrics specifics of safety-critical software development


Requirements allocated to Software
start
SW Requirements Analysis
Software Acquisition Process
Architectural Design
Design
Code and Unit Test
System Test, Verification, & Qualification
end

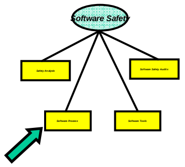
Technical data
Configuration Management
SQA
Verification
Validation
Joint Review
Audit
Problem Resolution
Estimating
Reuse

Risk Management
Metrics
SPEs
Software Release
Prototyping
Planning
SW Libraries
Defect Prevention
Process Improvement
SQM
Best Practices

Standards
Methods
Product

Software Safety Process

-- Software Process

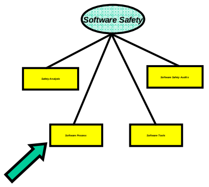


- **Identification and Standards . . .**
  - ***ID Software components to which safety processes apply***
  - ***ID Levels of criticality for each identified component***
  - ***ID and describe Architectural constraints***
    - Partitioning of software to nodes or address spaces
    - Processing resource 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

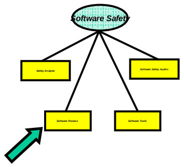


- **Software Methods**
  - ***Bi-directional Traceability of software safety requirements***
    - Requirements to design to code to test procedures
    - Test procedures to . . . requirements
  - ***Defect Prevention activities***
  - ***Joint review of software products involving application of safety practice***
  - ***Prototype software components built in support of safety-critical software development***
  - ***Test schedules and resources for safety-critical software***
  - ***Inspection or walkthrough review methods for each product involving safety-critical software***
  - ***Decision management process for reuse, use, and readiness of safety-critical software***
    - Including reuse of requirements, design, and test work products as well as multiple uses of 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

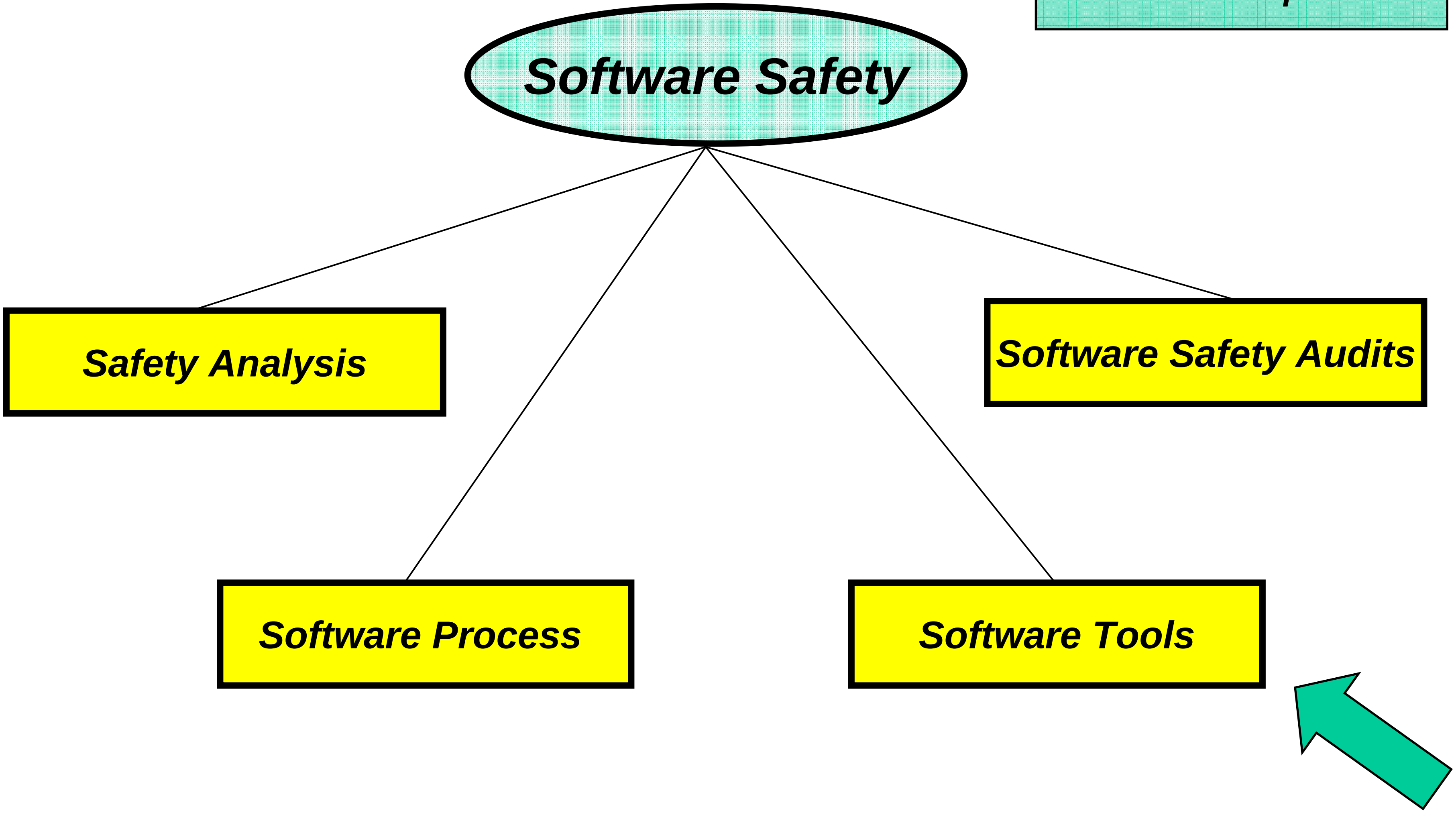


- **Software Product Assurance**
  - ***Mark requirements, design, code, and tests of safety-critical software***
  - ***Analysis and handling of dead code, deactivated code***
  - ***Verification of source in accordance with coding standards – automate checking, where practical***
    - Non-compliant software should be changed to be compliant or sufficient justification documented and reviewed by software team
  - ***Specify functional, structural coverage; complexity***
  - ***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***

Standards
Methods
Product

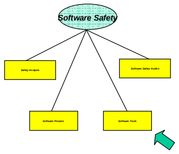
Software Safety Process

- Software Development
- Software Acquisition



Software Safety Process

-- Software Tools



Software Safety

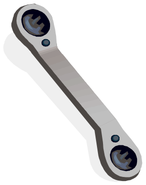
Safety Analysis

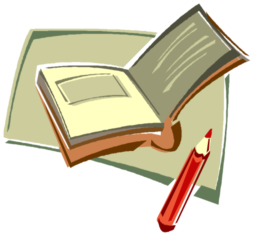
Software Safety Audits

Software Process

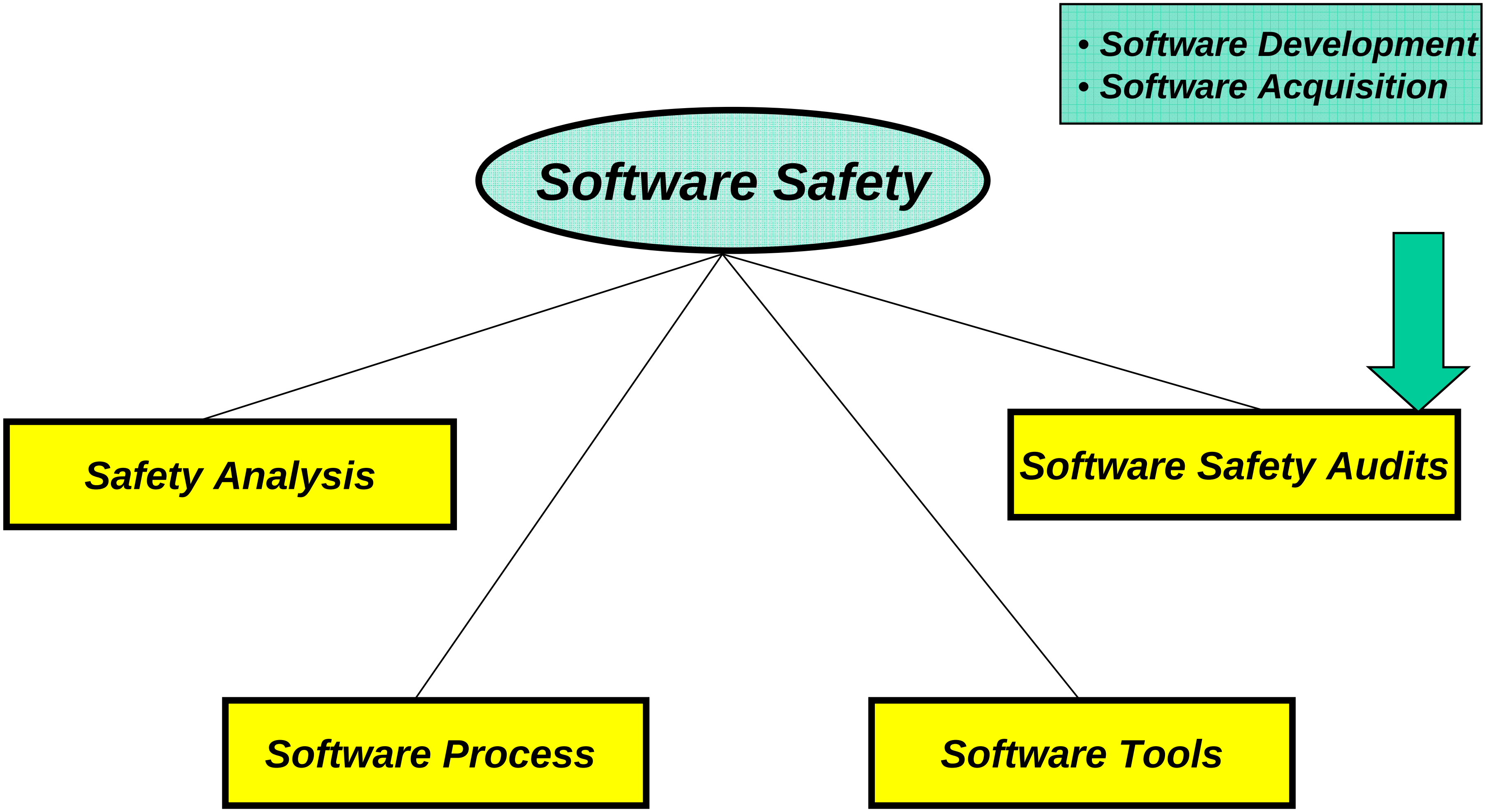
Software Tools

- **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*



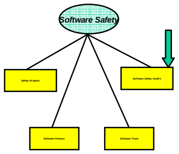


Software Safety Process

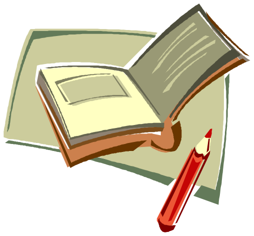


Software Safety Process

-- Software Safety Audits



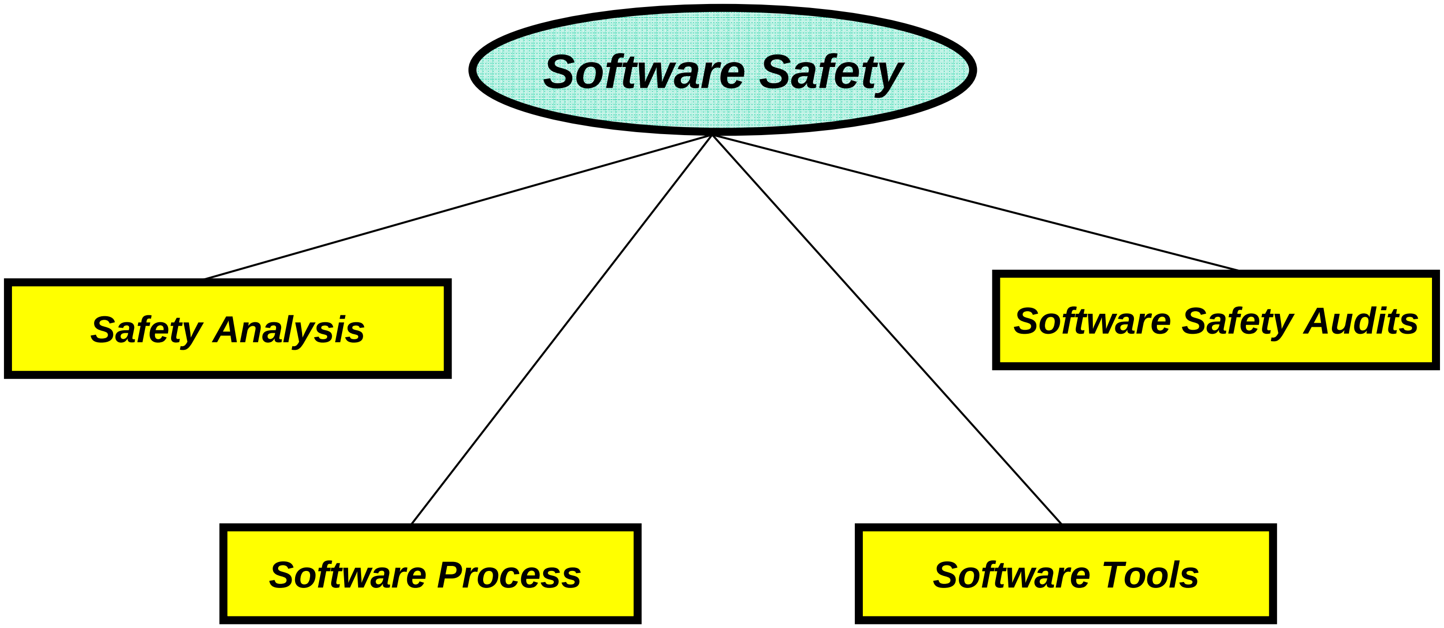
- **Software Safety Audits (also in SDP)**
  - ***Auditing provides some assurance for acquirer that contractors have built what they intended to build and it is of required quality***
  - ***Audits usually accomplished through sampled reviews of process work products (relative to the safety requirements and tasks)***
    - Variability in reviews dependant on auditor
  - ***SW 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

Safety Analysis

Software Safety Audits

Software Process

Software Tools

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 Plan***
    - 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
  - ***Review subcontractor data products to . . .***
    - Ensure production and control of required work products (i.e., evidence for audit) for safety-critical software development
  - ***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***

But wait . . . That’s not all ! !

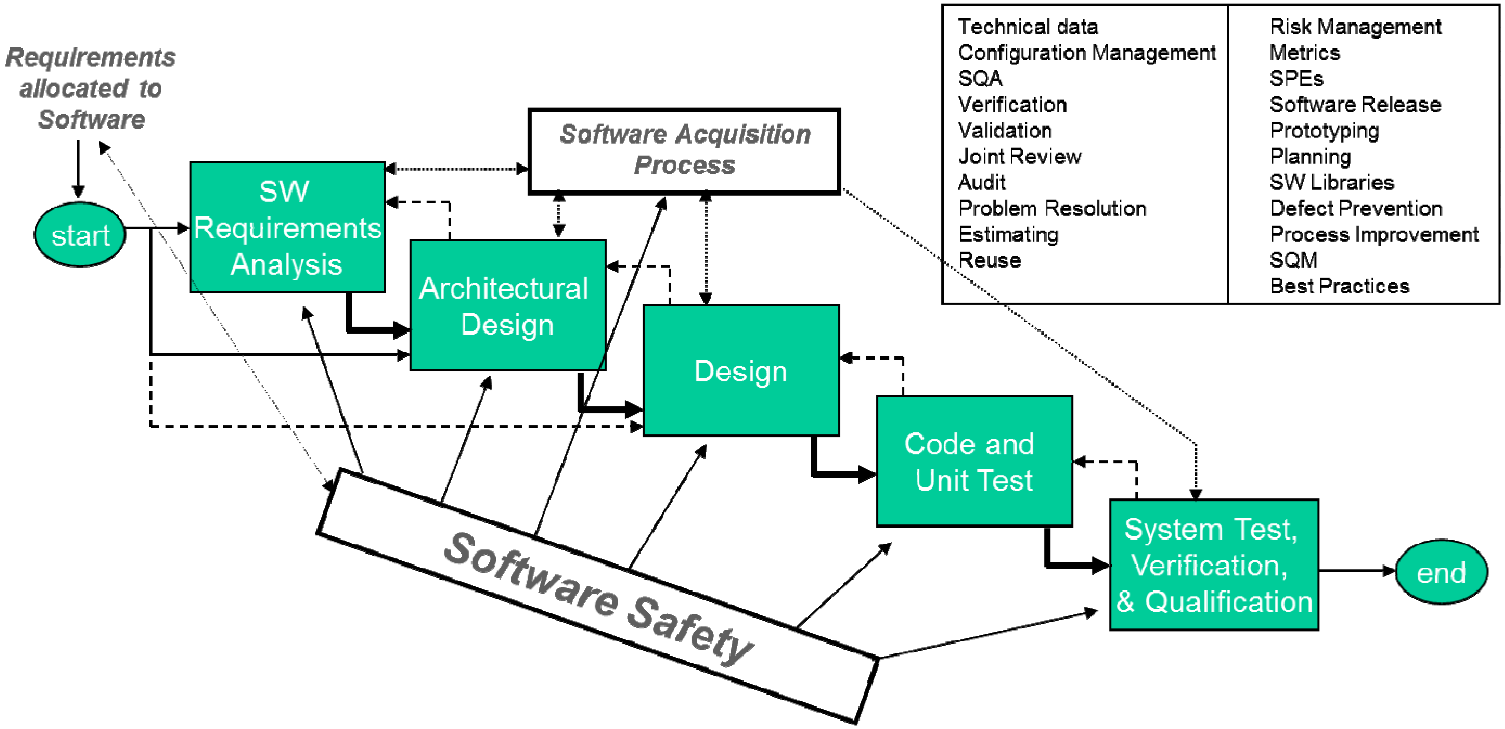
- **For highest levels of software assurance, may also require . . .**
  - ***Independence in verification activities***
  - ***Testing of every decision structure, every condition shown to take all possible outcomes at least once and each condition shown to affect outcome independently (MC/DC)***
  - ***Source to Object Correspondence***
    - Used when highest assurance required and compiler generates object 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 . . .**
  - ***Provide for independent oversight, collaboration, and verification***

Ultimately . . .

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



Requirements allocated to Software

start

SW Requirements Analysis

Software Acquisition Process

Architectural Design

Design

Code and Unit Test

System Test, Verification, & Qualification

end

Software Safety

Technical data
Configuration Management
SQA
Verification
Validation
Joint Review
Audit
Problem Resolution
Estimating
Reuse

Risk Management
Metrics
SPEs
Software Release
Prototyping
Planning
SW Libraries
Defect Prevention
Process Improvement
SQM
Best Practices

Whew!

Example

Software Safety Process

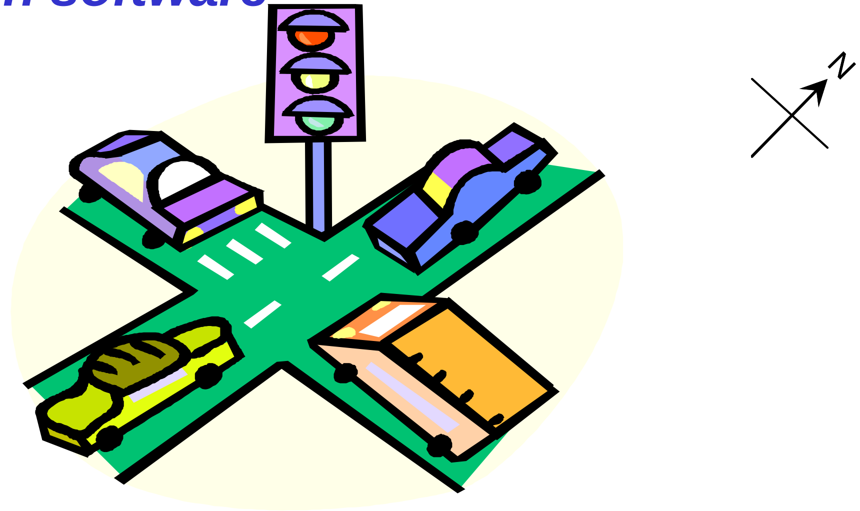
Tailoring Guidelines

		Project 1					Project 2	
		Safety Criticality Level					Safety Criticality Level	
Req. ID	**Software Process Requirement Text**	**A**	**B**	**C**	**D**	**E**	**S-C**	**Not S-C**
Software Development								
1.1.3.1-1	Evaluate the SDP requirements for development of software for safety-critical applications listed in the attached list of <u>SDP Requirements for Safety-Critical Software Development</u> for applicability to the software project and establish appropriate tailoring of these SDP requirements.	X	X	X	X		X	
1.1.3.1-2	Ensure that costs applicable to the development of identified safety-critical software are included in the software cost estimates. *Reference section 4.9 Software Estimating Practice for more information.*	X	X	X	X		X	
1.1.3.1-3	Provide the SDP to system safety and systems engineering organizations for review and coordination in addition to normal SDP review and approval requirements.	X	X	X			X	
1.1.3.1-4	Develop or adopt coding standards to be used for each language type used in development of safety-critical software.	X	X	X			X	
1.1.3.1-5	Assign the role of software safety engineer within the software development team. *This role is assigned by the SPM to an experienced and trained member of the software team. On small software teams, an appropriately trained SPM may fulfill this role.*	X	X	X			X	
1.1.3.1-6	Verify that software engineers have attended required software safety training courses prior to developing safety-critical software.	X	X	X			X	

Exercise

Exercise

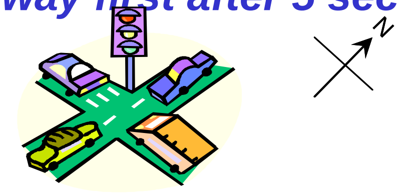
- **Real-world problem to understand application of software safety**
  - ***4-way Traffic Light at intersection of high-speed highways***
- **Exercise is to examine design of traffic light system, determine if software is safety-critical, and if so . . .**
  - ***Identify the levels of criticality and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



Exercise

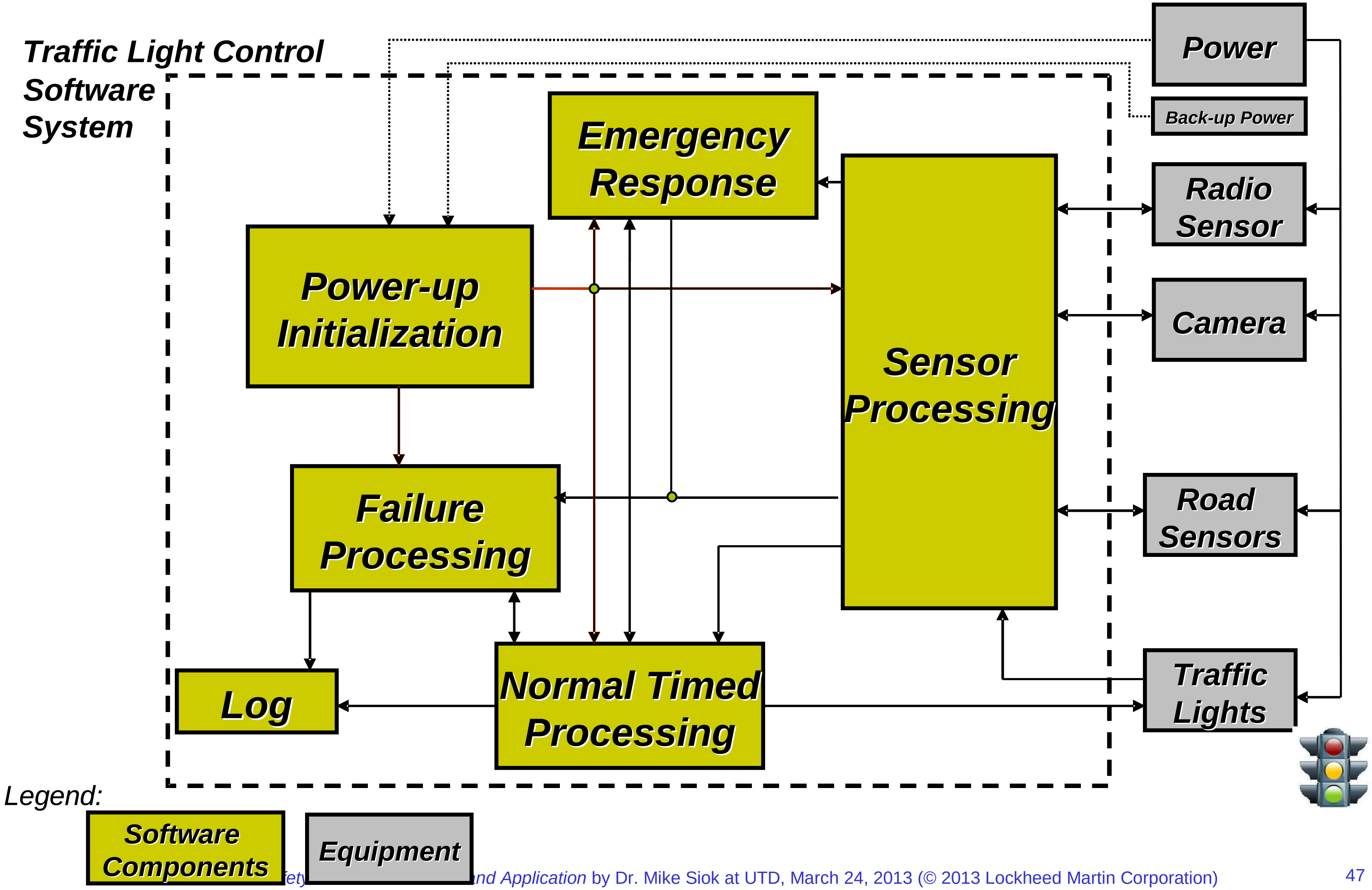
-- Requirements (Example)

- 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)



Traffic Light Control Software System

Power

Back-up Power

Radio Sensor

Camera

Road Sensors

Traffic Lights

Emergency Response

Power-up Initialization

Sensor Processing

Failure Processing

Log

Normal Timed Processing

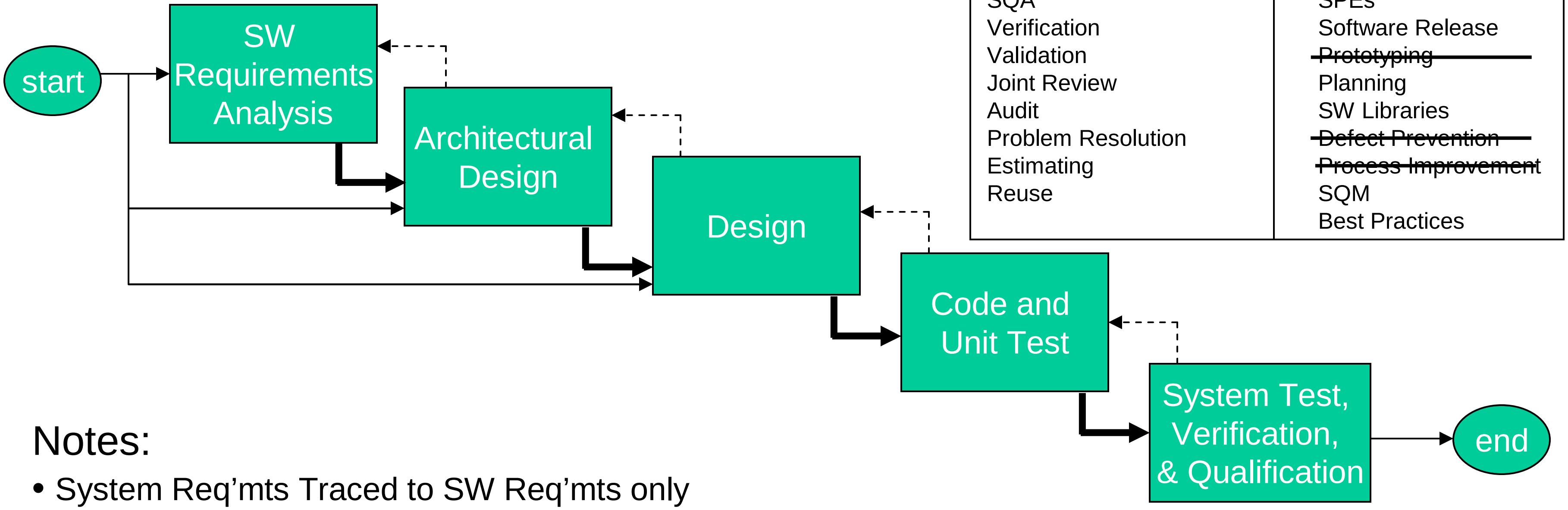
Legend:

Software Components

Equipment

Exercise

-- Current Software Process (Example)



Technical data
Configuration Management
SQA
Verification
Validation
Joint Review
Audit
Problem Resolution
Estimating
Reuse

Risk Management
Metrics
SPEs
Software Release
~~Prototyping~~
Planning
SW Libraries
~~Defect Prevention~~
~~Process Improvement~~
SQM
Best Practices

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:

Exercise

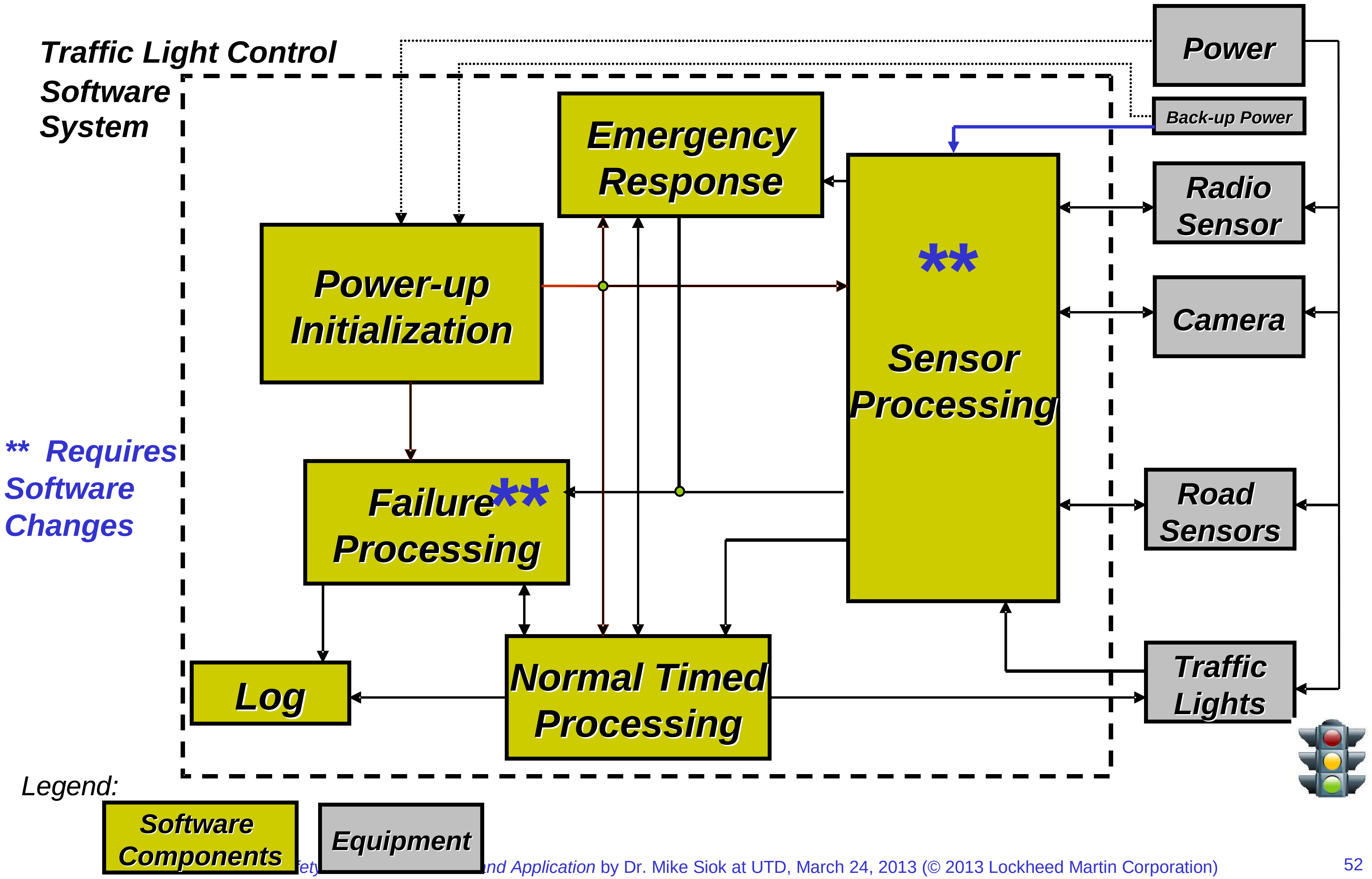
-- Determining Criticality . . . (example)

		Hazard Probability				
Hazard Severity Category		FREQUENT A	PROBABLE B	OCCASIONAL C	REMOTE D	IMPROBABLE E
CATASTROPHIC - Safety Critical event resulting in: death, aircraft loss or damage beyond economical repair; or severe environmental damage	I	HRI 1	HRI 2	[✓ HRI 4]	HRI 8	HRI 11
CRITICAL - Safety Critical event resulting in: Severe injury or occupational illness to any personnel that results in a permanent partial disability; aircraft or property damage > $1,000,000; a condition that requires immediate action to prevent the above (including Cat I) or major environmental damage.	II	HRI 3	HRI 5	HRI 6	HRI 10	HRI 15
MARGINAL - Minor injury or occupational illness; aircraft or property damage > $ 20,000; an inflight failure requiring termination of flight for safety reasons or correctable environmental damage.	III	HRI 7	HRI 9	HRI 12	HRI 14	HRI 17
NEGLIGIBLE - Less than minor injury or damage or minimal environmental damage. Mission can be continued with minimum risk.	IV	HRI 13	HRI 16	HRI 18	HRI 19	HRI 20

	MIL-STD-882D Hazard Severity Levels	*UK DEF-STAN-00-55 Software Safety Integrity Levels **	*RTCA/DO-178B Software Levels*	*Standard Model Software Criticality*
HRI 1 – 3	*I Catastrophic*	*SIL 4*	*A Catastrophic*	*Safety Critical*
HRI 4 – 7	*II Critical*	*SIL 3*	*B Hazardous*	*Safety Significant*
HRI 8 – 10	*III Marginal*	*SIL 2*	*C Major*	*Safety Related*
HRI 11 -- 20	*IV Negligible*	*SIL 1*	*D Minor*	*Minor Safety Impact*
	---	*SIL 1*	*E No Effect*	*No Safety Impact*

Exercise

-- System/Software Functional Block Diagram (Example)



Exercise

-- Software Safety Process Tailoring (Example)

Software Safety Process
Tailoring Guidelines

		Project 1					*Project 2*	
		Safety Criticality Level					Safety Criticality Level	
Req. ID	**Software Process Requirement Text**	**A**	**B**	**C**	**D**	**E**	**S-C**	**Not S-C**
Software Development								
1.1.3.1-1	Evaluate the SDP requirements for development of software for safety-critical applications listed in the attached list of SDP Requirements for Safety-Critical Software Development for applicability to the software project and establish appropriate tailoring of these SDP requirements.		X				X	
1.1.3.1-2	Ensure that costs applicable to the development of identified safety-critical software are included in the software cost estimates. *Reference section 4.9 Software Estimating Practice for more information.*		X				X	
1.1.3.1-3	Provide the SDP to system safety and systems engineering organizations for review and coordination in addition to normal SDP review and approval requirements.		X				X	
1.1.3.1-4	Develop or adopt coding standards to be used for each language type used in development of safety-critical software.		X				X	
1.1.3.1-5	Assign the role of software safety engineer within the software development team. *This role is assigned by the SPM to an experienced and trained member of the software team. On small software teams, an appropriately trained SPM may fulfill this role.*		X				X	
1.1.3.1-6	Verify that software engineers have attended required software safety training courses prior to developing safety-critical software.		X				X	

Exercise

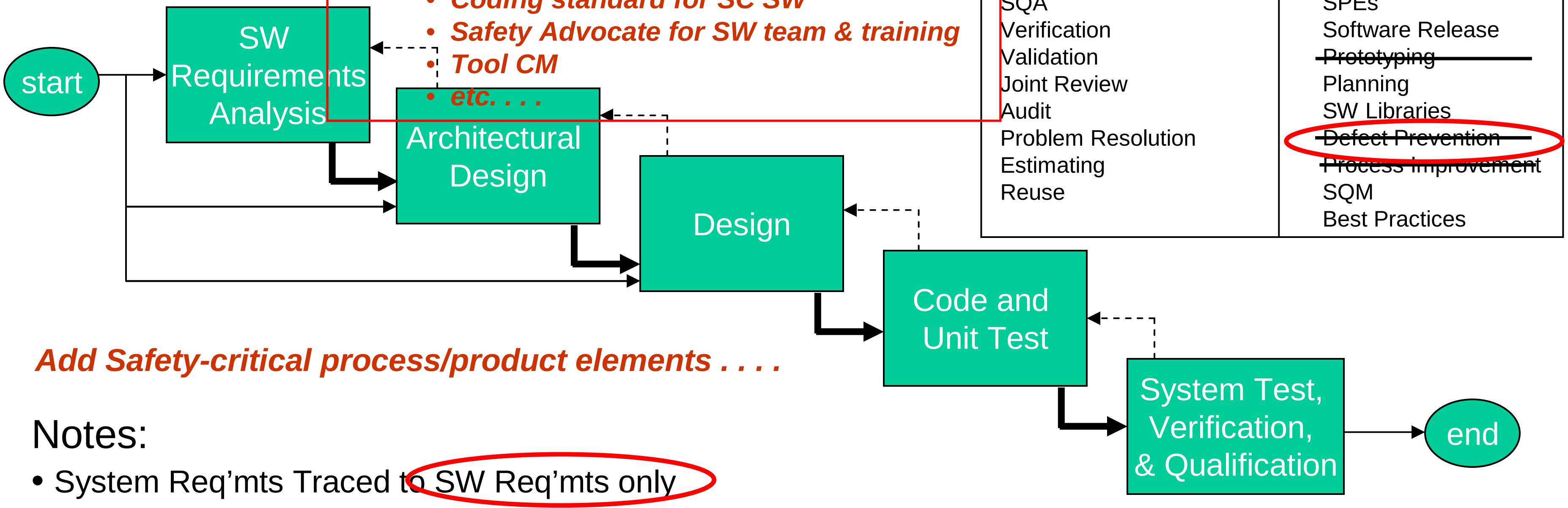
-- Software Process with Changes for Safety (Example)

Added Appendix/Section for Safety to SDP

- ***Addresses items in red and***
  - ***ID of SC SW & criticality***
  - ***Coding standard for SC SW***
  - ***Safety Advocate for SW team & training***
  - ***Tool CM***
  - ***etc. . . .***

Technical data
Configuration Management
SQA
Verification
Validation
Joint Review
Audit
Problem Resolution
Estimating
Reuse

Risk Management
Metrics
SPEs
Software Release
~~Prototyping~~
Planning
SW Libraries
~~Defect Prevention~~
~~Process Improvement~~
SQM
Best Practices



Add Safety-critical process/product elements . . . .

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	Project:: SW Safety Course		Date Opened:
Engineer: <name>	System: Traffic Light Example	Effectively:	Status: Open ○
	Subsystem: Power Subsystem	Initial Risk: Severity: __ Probability:__ Category:	In-Work ○
	Phase:	Modified Risk: Severity: __ Probability:__ Category:	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 HRI 4, DO-178B level B.

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:

Exercise

-- Determining Criticality After Controls . . . (example)

Hazard Severity Category		Hazard Probability FREQUENT A	PROBABLE B	OCCASIONAL C	REMOTE D	IMPROBABLE E
CATASTROPHIC - Safety Critical event resulting in: death, aircraft loss or damage beyond economical repair; or severe environmental damage	I	HRI 1	HRI 2	HRI 4	[✓ HRI 8]	HRI 11
CRITICAL - Safety Critical event resulting in: Severe injury or occupational illness to any personnel that results in a permanent partial disability; aircraft or property damage > $1,000,000; a condition that requires immediate action to prevent the above (including Cat I) or major environmental damage.	II	HRI 3	HRI 5	HRI 6	HRI 10	HRI 15
MARGINAL - Minor injury or occupational illness; aircraft or property damage > $ 20,000; an inflight failure requiring termination of flight for safety reasons or correctable environmental damage.	III	HRI 7	HRI 9	HRI 12	HRI 14	HRI 17
NEGL IGIBLE - Less than minor injury or damage or minimal environmental damage. Mission can be continued with minimum risk.	IV	HRI 13	HRI 16	HRI 18	HRI 19	HRI 20

	MIL-STD-882D Hazard Severity Levels	*UK DEF-STAN-00-55 Software Safety Integrity Levels* *	*RTCA/DO-178B Software Levels*	*Standard Model Software Criticality*
HRI 1 – 3	*I Catastrophic*	*SIL 4*	*A Catastrophic*	*Safety Critical*
HRI 4 – 7	*II Critical*	*SIL 3*	*B Hazardous*	*Safety Significant*
HRI 8 – 10	*III Marginal*	*SIL 2*	*C Major*	*Safety Related*
HRI 11 -- 20	*IV Negligible*	*SIL 1*	*D Minor*	*Minor Safety Impact*
	---	*SIL 1*	*E No Effect*	*No Safety Impact*

Exercise

Now it's your turn

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:

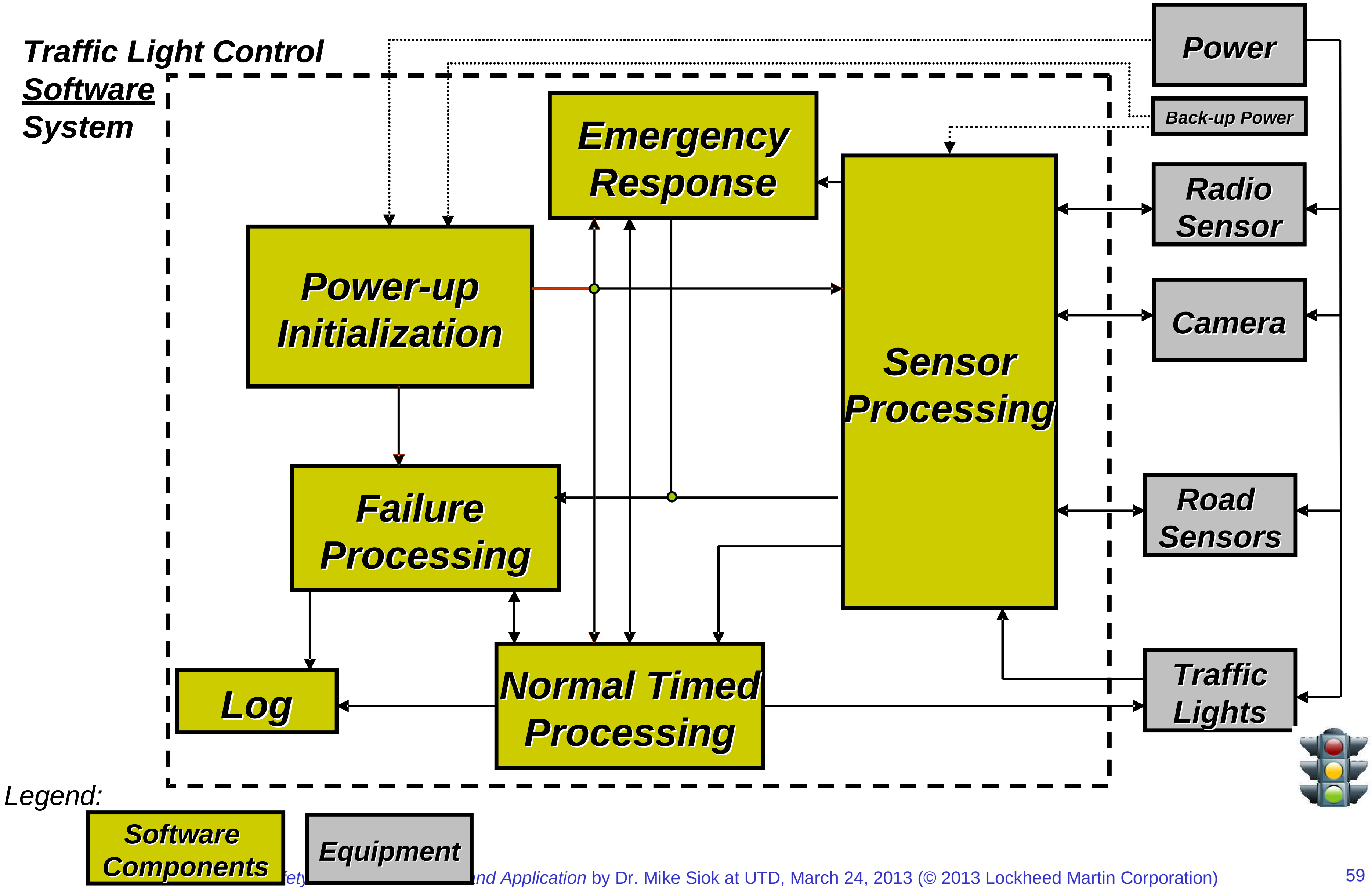
Effects after Controls:

Remarks:

Hazard Closure Evidence:

Exercise

-- System/Software Functional Block Diagram



Traffic Light Control Software System

Power

Back-up Power

Emergency Response

Power-up Initialization

Sensor Processing

Radio Sensor

Camera

Failure Processing

Road Sensors

Log

Normal Timed Processing

Traffic Lights

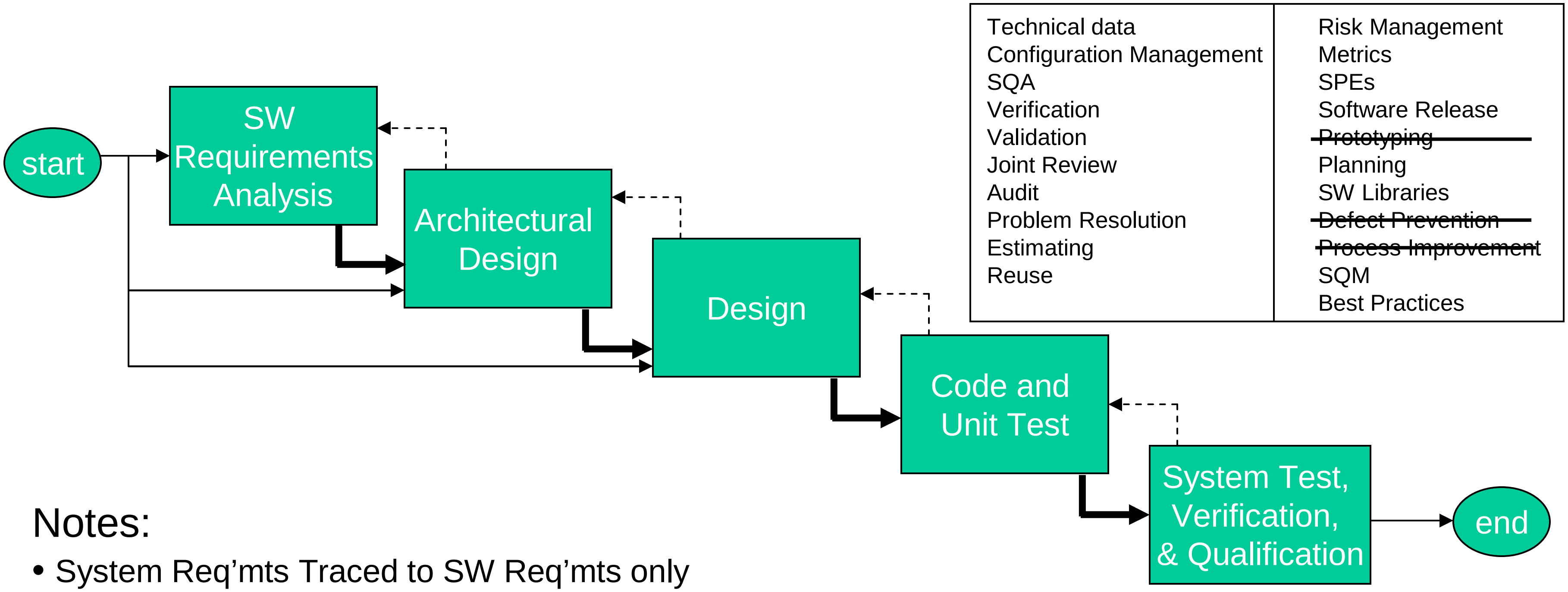
Legend:

Software Components

Equipment

Exercise

-- “Current” Software Process



- Technical data
- Configuration Management
- SQA
- Verification
- Validation
- Joint Review
- Audit
- Problem Resolution
- Estimating
- Reuse
- Risk Management
- Metrics
- SPEs
- Software Release
- ~~Prototyping~~
- Planning
- SW Libraries
- ~~Defect Prevention~~
- ~~Process Improvement~~
- SQM
- Best Practices

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

-- 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.

Exercise

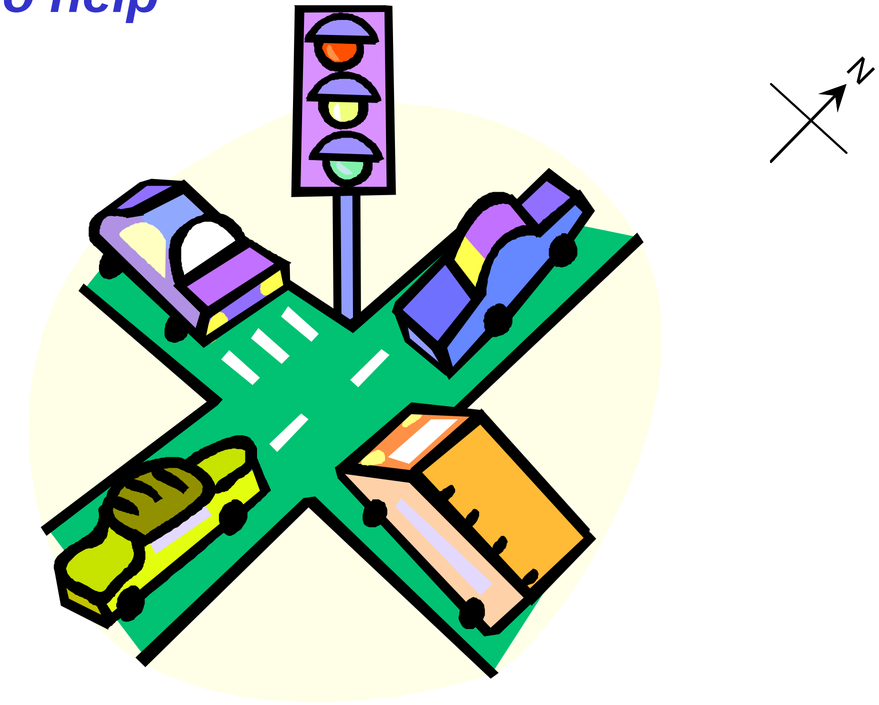
-- Exercise Hazards for Group Work

- 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.

Exercise Review

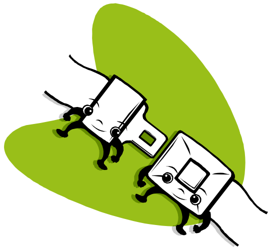
- You were to examine design of traffic light system, define hazard and control, determine if software was safety critical, identify levels of criticality, and why and modify software process accordingly
  - *Checklists were provided to help*

- Present solutions . . . .



Summary

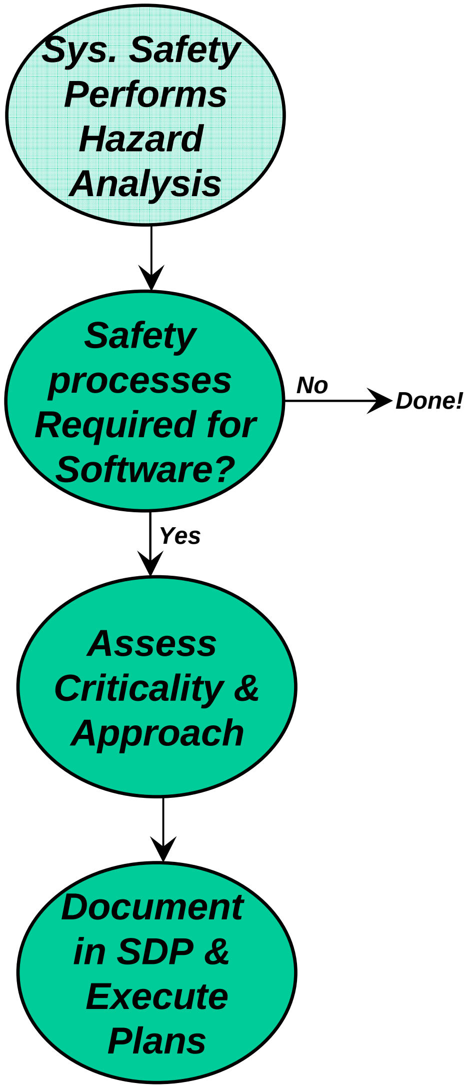
Summary



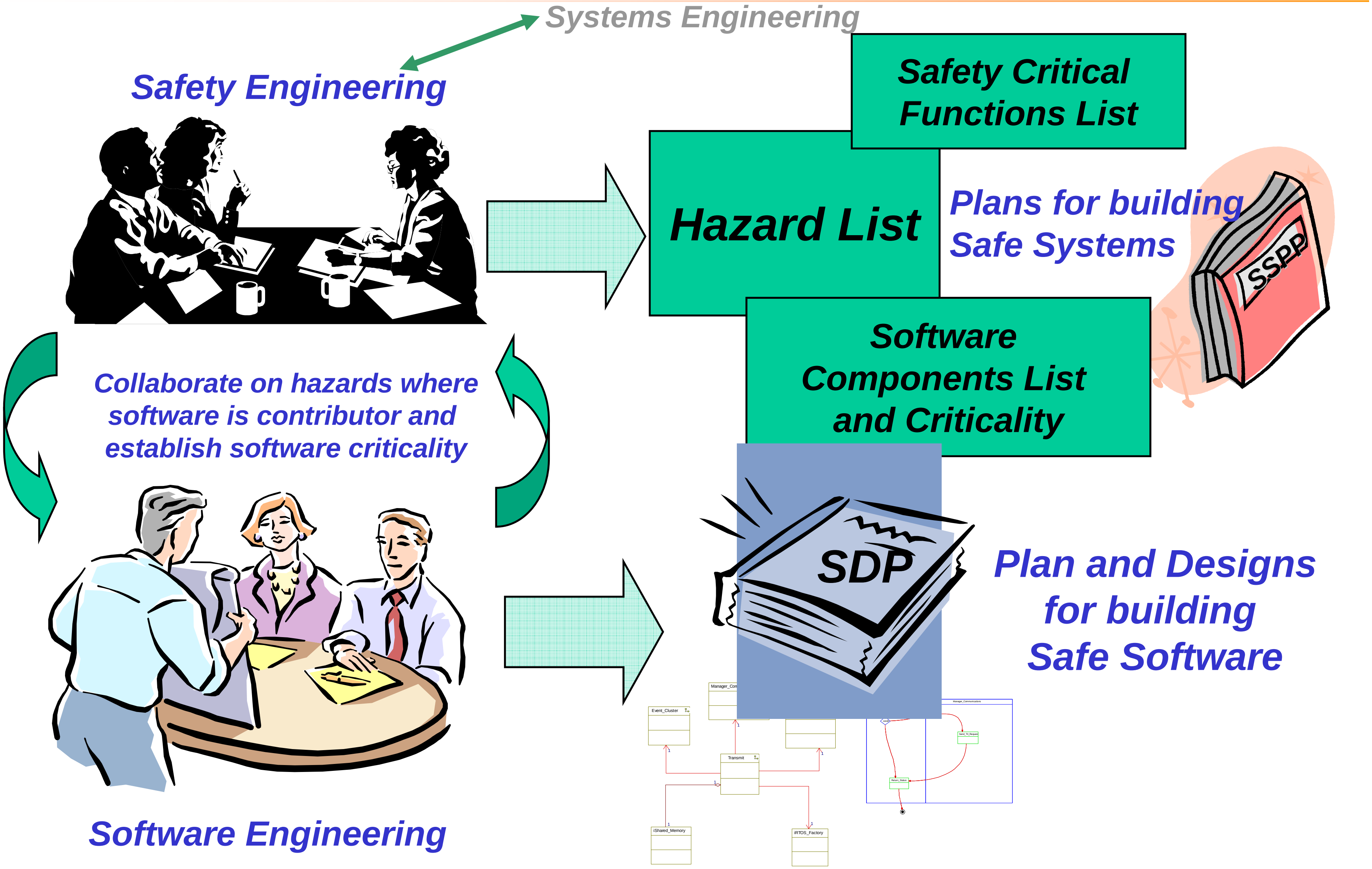
- Safe Software ≠
  - *Lower software defect rates*
  - *Reliable Software*
  - *Secure Software*
- Safety is a systems attribute
  - *Software Engineering and software are contributors to safe systems and safe operations*
- Safety Engineering conducts hazard analysis on program
  - *Software Engineering 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 . . .
  - *In-house developed software*
  - *Acquired software*
- Software Engineering Process Manual documents Software Safety Practice for LM Aero
  - *Context for LM Aero product software*
  - *Process requirements*
  - *Process Tailoring Guidance*
- Software Safety process should be tailored to specific program application
  - *Tailoring guidance provided and available*


Sys. Safety Performs Hazard Analysis
Safety processes Required for Software?
No
Done!
Yes
Assess Criticality & Approach
Document in SDP & Execute Plans

Summary (Cont'd)



Systems Engineering

Safety Engineering

Safety Critical Functions List

Hazard List

Plans for building Safe Systems

Software Components List and Criticality

Collaborate on hazards where software is contributor and establish software criticality

Plan and Designs for building Safe Software

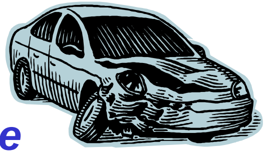
Software Engineering

Software Failures Affect US

… a few more recent examples and last reminders

- **Software Glitch Delayed Release of Results**
  - ***(Sept. 8, '04, Las Vegas, NV) -- For the second time during a busy election, the county's election department is plagued with problems. The Registrar of Voters says that software mishap was just one problem they had to deal with and it must be fixed before the general election.***

- **Ford Recalls F150 and Lincoln Mark LT trucks for Brake Errors (2006)**
  - ***Ford and Lincoln Mercury are recalling over 211,000 2006 Ford F-150 and 2006 Lincoln Mark LT trucks because a software glitch can disable the ABS brake warning system light if the system becomes inoperative.***



- **Prius Problems Traced to Software Glitch**
  - **(May 18, 2005)** ***--Toyota Motor Corp is focusing efforts on a software problem in the popular hybrid Prius automobile after complaints that the gas-electric hybrid cars stall or shut down without warning while driving at highway speeds (2004 and 2005 model cars).***

- **Nissan Leaf recalled for SW**
  - ***2011 -- Nissan Motor Co. is recalling 5,300 Leaf electric cars back to dealerships to fix a software glitch that can keep it from starting. Reprogram engine controller.***

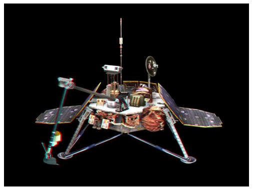
Software Failures Affect US

… a few more examples and last reminders

- Mars Global Surveyer
  - *Initiating event - two bad addresses uploaded while in orbit*
    - Software exceeded limits, locking solar panel gimbals
    - As designed, MGS reoriented itself to face panels toward sun
    - Battery on sun side overheated
    - Power management software design ‘assumed’ that battery overheating was due to overcharging and commanded charging system shutdown
  - *Vehicle was lost*

1996

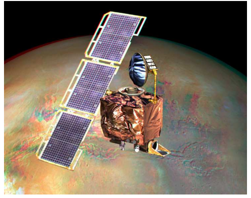
- Mars Polar Lander
  - *On final descent, landing strut deployed as planned caused sensor vibrations*
    - Software misinterpreted vibration-induced signals from accelerometers as touchdown
  - *Software subsequently shut down the descent engine about 40 meters above the Martian surface*
  - *Hard landing, vehicle lost*



1999

- Mars Climate Orbitor
  - *During transit to Mars, units discrepancy (lb-secs vs. newton-secs) in software undiscovered*
    - Data was loaded into tables in units different from software input expectation
  - *In-transit trajectory errors accumulated, putting the vehicle too close to planet for orbit insertion burn*
  - *Vehicle lost*

1999



Software Failures Affect US

… a few more examples and last reminders

- **Mishaps where software-related problems were reported to play a 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*
2001	*5*	*Crash of V-22 Osprey tilt-rotor helicopter caused by software anomaly*
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 Reference . . .

- **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-882D – Standard Practice for System Safety***
- ***RTCA, Inc., DO-178B, Software Considerations in Airborne Systems and Equipment Certification***
- ***RTCA, Inc., DO-248B, Final report for Clarification of DO-178B***
- **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 8604
Fort Worth, TX 76101
Tel: (817) 935-4514
FAX: (817) 762-9428
Email: Mike.F.Siok@lmco.com

Lockheed Martin Aeronautics Company

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

Lockheed Martin Aeronautics Company