Lecture Description

- An introduction to software safety
  - based on existing government and commercial standards
  - explains what software safety means,
  - how software can contribute to safety problems, and
  - what techniques are used to deal with safety-critical software
Why Be Concerned about Software Safety

- Can Software Harm Anyone?

Eek!!!

It's Software!

Software by itself seems pretty innocuous – but ...

Ways that Software can be Harmful

- It can Control the Behavior of Dangerous Devices
  - Robots
  - Weapons
  - Security Doors at Building entry
  - Medical Devices
  - Chemical Experiments
  - Factory Manufacturing Lines
  - …

- It can Send Information to People who do Potentially Dangerous Things
  - Location of Airplanes for ATC
  - Identification of Intruders
  - …

- It can Deceive
  - Internet scams

- And on and on …
Software is Often Used for Jobs that Once Called for Human Judgment

Software Safety Starts with System Safety

- Software is always part of a system
  - A database
  - A network
  - A vehicle
  - …
- If the system can harm someone, then the software may be a factor in whether the system harms someone
- So we have to start by analyzing the safety issues of the system
Basic System Safety Process

- Identify the Potential Hazards
- Decompose Hazard Threads through Subsystem Components, including SW
- Link/Trace to Requirements
- Generate Appropriate Mitigation Strategy
- Implement the Mitigation
- Verify that the Mitigation is Implemented and that it Functions as Expected
- Document Safety Artifacts

Fault Tree Concept

Keep going down until you reach a point where you can do something about it.
An Example of a System Level Hazard List

- Uncontrolled explosion
- Uncontrolled fire
- Injury and/or illness
- Blockage of ingress/egress paths
- Structural failure
- Collision
- Uncontrolled activation of ordinance
- Electromagnetic interference
- Hazardous/reactive materials
- Electrical energy
- Improper engagement control
- Surface/air contamination
- Corrosion resulting in loss of strength or integrity of exposed surfaces
- Batteries (exposure to toxic material, explosion)
- Radiation (ionizing and non-ionizing)
- Uncontrolled/unsupervised robotic operations

Notice how often the word “control” is used!

- Software is often used to replace mechanical or human controls.

- When software controls something it is often potentially safety critical.

- But that’s not the only way software can be safety critical.
Some Software Functions that May be Safety Critical

- Assessment of overall system health (e.g., power-up and run-time monitors, heartbeat, program memory CRCs, range checks, CPU health)
- Enforcement of critical timing
- Exception handling including, failure/malfunction detection, isolation, and containment
- Functions that execute the system’s response to detected failures/malfunctions
- Functions that enhance the system’s survivability (preservation of core functionality)
- Data quarantine/sanitization
- Range and reasonableness (sanity) checking
- Tamper and cyber-attack proofing
- Authentication for lethal actions
- Inhibiting functions and interlocks

Example Software Contribution to Hazard (1/3)

<table>
<thead>
<tr>
<th>Example SW Contribution</th>
<th>Symptom</th>
<th>Example Potential Cause</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incorrect Safety-Critical Alerts and Warnings.</td>
<td>Safety Critical alerts are incorrect, or are not triggered by Safety Critical Events. Alerts fail to warn the user of an unsafe condition, and/or an Unsafe System State.</td>
<td>Software design fails to account for the functionality to alert the operator to unsafe condition and/or state.</td>
</tr>
<tr>
<td>Incorrect Data Transfer Messages (transmit and receive).</td>
<td>Data transferred in the wrong format and the Safety Critical Data are interpreted incorrectly.</td>
<td>Failure to validate data transfer with the appropriate parity, check sums to validate Safety Critical data.</td>
</tr>
</tbody>
</table>
### Example Software Contribution to Hazard (2/3)

<table>
<thead>
<tr>
<th>Example SW Contribution</th>
<th>Symptom</th>
<th>Example Potential Cause</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Storage Failures (Safety Critical Data corrupted and or lost)</td>
<td>Safety-Critical Displays are confusing, and/or incorrect in presenting Safety Critical Data.</td>
<td>Safety critical data were not properly checksummed; data overwritten by mistake</td>
</tr>
<tr>
<td>Incorrect data transfer between processors</td>
<td>Incorrect message received</td>
<td>Failure to perform verification checks in both processors prior to transferring Safety Critical data.</td>
</tr>
</tbody>
</table>

### Example Software Contribution to Hazard (3/3)

<table>
<thead>
<tr>
<th>Example SW Contribution</th>
<th>Description</th>
<th>Example Potential Cause</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timing and Interrupt failures</td>
<td>Interrupts occur at the wrong time and potentially interrupt a Safety Critical process, or introduce a potential hazard.</td>
<td>Interrupts are out of synch with system time, and/or interrupt a Safety Critical Process/Path which introduces a potential hazard.</td>
</tr>
<tr>
<td>Incorrect Modes</td>
<td>Software signals the system to fire a weapon when it should not.</td>
<td>Switch from training to “live” mode is not correctly reflected in the “mode” variable.</td>
</tr>
</tbody>
</table>
So What Should We Do About Safety Critical Software?

- Give up and let’s just go home!
- Mitigate the Risks

Potential Software Risk Mitigation Strategies

- Wrappers to Constrain Inputs/Outputs
- Timers/Heartbeats/Counters
- Fault Trapping
- Error Trapping
- Command Retry
- Parity/Checksums
- Redundancy/Voting
- Monitoring
- Synchronization/Timing
- Timeout
- Reasonableness Procedures
- Interlocks/Inhibits
- Range/Sequence Checking
- Transition Checking
- Hysteresis for Man-Machine Interactions to allow time for Human Interpretation
- Partitioning/Isolation
- Redundant but Dissimilar Algorithms
- Functional Separation
- Failsafe Strategies
- Fault/Error Tolerant Strategies

Not a Comprehensive List
We will discuss some of these techniques in a later lecture
How Seriously Must we Treat Potential System Safety Issues?

- **Mil-STD 882D**
- **Hazard Probability** (how likely is the hazard?)
- **Severity of Hazard** (how much damage can be caused?)
- **Hazard Response Level or Risk Index** (how extensively our product development techniques must be enhanced)

This can be a very subjective evaluation. Any hazard should be avoided, but some are worse than others. The safety community for any application domain will, by and large, agree on a set of severity categories (see next slide).
Hazard Severity Categories Commonly Used in Military Safety Standards

- **Catastrophic**
  - could result in death, permanent total disability, loss exceeding $1M, or severe environmental damage violating law or regulation

- **Critical**
  - could result in permanent partial disability, injuries or illness affecting at least 3 people, loss exceeding $200K, or reversible damage to environment violating law or regulation

- **Marginal**
  - could result in injury or illness resulting in loss of 1 or more work days, loss exceeding $10K, or 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 exceeding $2K, or minimal environmental damage not violating law or regulations

---

How Seriously Must we Treat Potential **System** Safety Issues?

**Mil-STD 882D**

- **Hazard Probability** (how likely is the hazard?)

- **Severity of Hazard** (how much damage can be caused?)

- **Hazard Response Level or Risk Index** (how extensively our product development techniques must be enhanced)
### Typical Hazard Probabilities

<table>
<thead>
<tr>
<th>Level</th>
<th>Description</th>
<th>Description</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Frequent</td>
<td>Will Occur</td>
<td>1 in 100</td>
</tr>
<tr>
<td>B</td>
<td>Probable</td>
<td>Likely to Occur</td>
<td>1 in 1000</td>
</tr>
<tr>
<td>C</td>
<td>Occasional</td>
<td>Unlikely but Possible</td>
<td>1 in 10,000</td>
</tr>
<tr>
<td>D</td>
<td>Remote</td>
<td>Very Unlikely</td>
<td>1 in 100,000</td>
</tr>
<tr>
<td>E</td>
<td>Improbable</td>
<td>Can Assume Will Not Occur</td>
<td>1 in 1,000,000</td>
</tr>
</tbody>
</table>

Probability Level is Used to Select Appropriate Mitigation Actions

### Hazard Risk Index Combines Severity and Probability

#### Typical Hazard Risk Index Calculation

<table>
<thead>
<tr>
<th>Probability</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I - Catastrophic</td>
<td>U</td>
<td>U</td>
<td>M</td>
<td>M</td>
<td>N</td>
</tr>
<tr>
<td>II - Critical</td>
<td>U</td>
<td>U</td>
<td>M</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>III - Marginal</td>
<td>U</td>
<td>M</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>IV - Negligible</td>
<td>M</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>U - Unacceptable</th>
<th>Mitigate at High Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>M – Marginal Risk</td>
<td>Mitigate at Moderate Cost</td>
</tr>
<tr>
<td>N – No Significant Risk</td>
<td>Mitigate at Low Cost</td>
</tr>
</tbody>
</table>
One can deduce the probability of hardware failure by examining the properties of materials, laws of physics, data on material fatigue, etc.

But one cannot deduce the probability of software failure in such a manner.

For software, most safety experts use a different approach (next slide).

---

How Seriously Must we Treat Potential **Software** Safety Issues?

- **Severity of Hazard** (how much damage can be caused?)
- **Level of Control** (to what degree is software responsible?)
- **Software Level of Rigor** (how extensively our software development techniques must be enhanced)
The Software Level of Control is a measure of the degree to which software is responsible for the behavior of a system or of a specific system action that may lead to a safety hazard.

The higher the level of control, the more rigorous the process for software development.

---

An Example of Software Levels of Control (1/3)

- **I – Autonomous/Time Critical**
  - *Software exercises autonomous control* over potentially hazardous hardware systems, subsystems, or components without the possibility of intervention to preclude the occurrence of a hazard.
  - Failure of the software or a failure to prevent an event leads directly to a hazard's occurrence. This implies *no other failure is required to cause the hazard*.

- **IIa - Autonomous/Not Time Critical**
  - *Software exercises control* over potentially hazardous hardware systems, subsystems, or components *and allows time for intervention* by independent safety systems to mitigate the hazard. This implies that corrective action is possible and a second fault or error is required for this hazard to occur.
An Example of Software Levels of Control (2/3)

IIb Information Time Critical
- **Software displays information requiring immediate operator action** to mitigate a hazard. Software failures will allow or fail to prevent the occurrence of a hazard. This implies that *the operator is made aware of the existence of the hazard and intervention is possible.*

IIIa Operator Controlled
- **Software issues commands** over potentially hazardous hardware system subsystems or components *requiring human action to complete* the control function. *There are several redundant, independent safety measures for each hazardous event.*

IIIb Information Decision
- **Software generates information** of a safety-critical nature used to make safety-critical decisions. There are *several redundant, independent safety measures for each hazardous event.*

IV Not Safety Software
- **Software does not control safety-critical hardware systems**, subsystems, or components and *does not provide safety-critical information.*
Summary of Software Levels of Control

A software fault may:
- result directly in a hazard (LOC I),
- significantly impact the margin of safety (LOC IIa or IIb), but not result directly in a hazard, or
- have a minor impact on the margin of safety for a hazard (LOC IIIa or IIIb).

### Software Level of Rigor (how extensively our SW development techniques must be enhanced)

<table>
<thead>
<tr>
<th>Hazard Severity</th>
<th>I - Autonomous / Time Critical</th>
<th>IIb - Information / Time Critical</th>
<th>IIa - Autonomous s/ Not Time Critical</th>
<th>IIIa - Operator Controlled</th>
<th>IIIb - Information Decision</th>
<th>IV - Not Safety Software</th>
</tr>
</thead>
<tbody>
<tr>
<td>I - Catastrophic</td>
<td>LOR-3</td>
<td>LOR-3</td>
<td>LOR-2</td>
<td>LOR-2</td>
<td>LOR-2</td>
<td>N/A</td>
</tr>
<tr>
<td>II - Critical</td>
<td>LOR-3</td>
<td>LOR-2</td>
<td>LOR-2</td>
<td>LOR-2</td>
<td>LOR-2</td>
<td>N/A</td>
</tr>
<tr>
<td>III - Marginal</td>
<td>LOR-2</td>
<td>LOR-2</td>
<td>LOR-2</td>
<td>LOR-1</td>
<td>LOR-1</td>
<td>N/A</td>
</tr>
<tr>
<td>IV - Negligible</td>
<td>LOR-1</td>
<td>LOR-1</td>
<td>LOR-1</td>
<td>LOR-1</td>
<td>LOR-1</td>
<td>N/A</td>
</tr>
</tbody>
</table>
Techniques for Level of Rigor 1

**INPUT**
- SAFETY CRITICAL FUNCTIONS
- TOP LEVEL MISHAPS
- REQUIREMENTS & DESIGN CHANGES
- APPLICABLE STANDARDS (e.g., STANAG 4404)
- LESSONS LEARNED

**EXTENT OF ANALYSIS**
- PERFORM REQUIREMENTS ANALYSIS
- DEVELOP TEST PLAN & PROCEDURES
- OPERATIONS & TEST ANALYSIS

**PRODUCTS**
- Software Safety Requirements
- Criticality Matrix
- Traceability Matrix
- Preliminary Hazard Analysis
- Safety Requirements/ Criteria Analysis
- Hazard Control Records
- Computer Program Change Requests

**Techniques for Level of Rigor 2**

**INPUT**
- SAFETY CRITICAL FUNCTIONS
- TOP LEVEL MISHAPS
- REQUIREMENTS & DESIGN CHANGES
- APPLICABLE STANDARDS (e.g., STANAG 4404)
- LESSONS LEARNED

**EXTENT OF ANALYSIS**
- PERFORM REQUIREMENTS ANALYSIS
- DEVELOP TEST PLAN & PROCEDURES
- OPERATIONS & TEST ANALYSIS

**PRODUCTS**
- Software Safety Requirements
- Criticality Matrix
- Traceability Matrix
- Preliminary Hazard Analysis
- Safety Requirements/ Criteria Analysis
- Hazard Control Records
- Computer Program Change Requests

- Criticality Matrix Update
- Traceability Matrix Update
- Safety Test Requirements
- Subsystem Hazard Analysis
- Hazard Control Records
- Computer Program Change Requests

- System Hazard Analysis
- Operating & Support Hazard Analysis
- Safety Assessment Report
- Safety Test Verification Report
- Hazard Control Records
- Computer Program Change Requests
Techniques for Level of Rigor 3

INPUT

SAFETY CRITICAL FUNCTIONS

TOP LEVEL MISHAPS

REQUIREMENTS & DESIGN CHANGES

APPLICABLE STANDARDS (e.g., STANAG 4404)

LESSONS LEARNED

EXTENT OF ANALYSIS

PERFORM REQUIREMENTS ANALYSIS

SR/CA

DEVELOP TEST PLAN & PROCEDURES

OPERATIONS & TEST ANALYSIS

DESIGN ANALYSIS

CODE ANALYSIS

REQUIREMENTS COMPLIANCE

PRODUCTS

SOFTWARE SAFETY REQUIREMENTS

CRITICALITY MATRIX

TRACEABILITY MATRIX

PRELIMINARY HAZARD ANALYSIS

SAFETY REQUIREMENTS/CRITERIA ANALYSIS

HAZARD CONTROL RECORDS

COMPUTER PROGRAM CHANGE REQUESTS

CRITICALITY MATRIX UPDATE

TRACEABILITY MATRIX UPDATE

SUBSYSTEM HAZARD ANALYSIS

HAZARD CONTROL RECORDS

COMPUTER PROGRAM CHANGE REQUESTS

SYSTEM HAZARD ANALYSIS

HAZARD CONTROL RECORDS

COMPUTER PROGRAM CHANGE REQUESTS

OPERATING & SUPPORT HAZARD ANALYSIS

SAFETY ASSESSMENT REPORT

SAFETY TEST VERIFICATION REPORT

HAZARD CONTROL RECORDS

COMPUTER PROGRAM CHANGE REQUESTS