Using Measures and Risk Indicators for Early Insight Into Software Product Characteristics such as Software Safety

Victor R. Basili
March 28, 2013
Outline

1. Emergent properties, process risk, and the Process Risk Indicator (PRI) method

2. Example emergent property: Software Safety

3. Applying PRI to measure software safety risk to the NASA Constellation Project

4. Lessons learned and recommendations
Introduction

Non-functional requirements are major cost drivers in the development of large systems in the aerospace, defense, energy and related industries

- Non-functional requirements describe desired *emergent properties*, such as safety, reliability, security and performance.
- These *emergent properties* can only be fully tested when the system is complete.
- Corrective actions after testing or operations are costly to implement.

In systems development, we ensure the desired emergent properties will be present in our system by applying *specific processes*

- **Security**: threat modeling, penetration testing
- **Safety**: hazard analysis, fault-tree analysis, STAMP
- **Reliability**: probabilistic risk assessment, Failure Modes and Effects Analysis
The Challenge – Process Risk

The processes for achieving an emergent property must satisfy three assumptions:

1. The process is capable of achieving the property or mitigating the risk of not achieving the property;
2. The process is appropriate for the development context;
3. The process is followed correctly

If a process fails to meet any of these three assumptions there is a risk that the product will not achieve the desired property.
Types of Risk

*Product or technical risk* is the risk that a system will not achieve a desired property, such as functionality, reliability, performance, safety or security.

*Process risk* is the risk created by the (correct or incorrect) application of a process that leads to a product risk.
Mitigating types of risk

Managing *product risks* associated with emergent properties is challenging because these properties are a function of the entire system

- Hard to test these properties before the system is completed
- Want to address **risk of not achieving** them as early as possible

Mitigating *process risks* often takes the form of ensuring that the correct process is followed

- Quality assurance, process conformance evaluations, application of process improvement framework
- Is the process is flawed, not followed or the context inappropriate

How do we ensure that the process is correct, appropriate, and being followed?
Process Risk Indicator (PRI) Key Points

• Focuses on the syntactic and semantic content of process artifacts

• Can be applied early in the life cycle to measure, track, and respond to risks that an emergent property will not be achieved

• Concept of risk is not traditional, i.e., a probability. It is a quantitative/qualitative identification of problems that can cause a risk of not achieving the property
Relationship between process and outputs

$\text{Property}^f (\text{Product}) = F^f (\text{Requirements, Process, People, Context, ...})$
Process Artifact Information

• **syntactic information**: data elements and their expected compositional format
  – the syntactic information of a defect report may include the description of the defect, the reproduction steps, and criticality of the defect
  – If these elements missing, there is risk that the defect tracking process is not being performed correctly.

• **semantic information**: meaning or interpretation of the syntactic data in the context of development
  – Is the right information in the defect report?
  – Interpreting semantic information frequently requires human expertise
Questions to identify and measure sources of process risk due to non-conformance, flaws in the process, or the applicability of the process to the context:

1. Is there any information in the process artifacts?
2. Is there enough information to perform a syntactic analysis of the data?
3. Is there enough information to perform a semantic analysis of the data?
4. Is the data semantically correct?

Positive answers to each successive question provides greater insights into the development process with a deeper understanding of the risks that may be present.
## Process Risk Indicator Method

PRI is our 6-step safety process risk approach applicable to emergent system properties, e.g. safety, reliability, …

| I. Identifying insight opportunities | 1. **Identify insight areas** from the development process that provide insight into risk areas.  
2. **Identify measurement opportunities** that provide insight into each risk area. |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>II. Evaluating the quality of information</td>
<td>3. <strong>Develop readiness assessment questions</strong> to identify if it is possible to delve deeper into the area.</td>
</tr>
</tbody>
</table>
| III. Measuring, interpreting, and providing advice | 4. **Define goals, questions, and measures for each risk area** to expose risks associated with process artifacts.  
5. **Develop and enumerate models** of how the measures will be interpreted via threshold values.  
6. **Propose responses to identified risks**, e.g., decisions and actions. |
Outline

1. Emergent properties, process risk, and the Process Risk Indicator (PRI) method

2. Example emergent property: Software Safety

3. Applying the PRI to the NASA Constellation Project to measure software safety risk

4. Lessons learned and recommendations
Software safety risk – an example emergent property

A safety risk is a hazard, which is any real or potential condition that can cause:

– injury, illness, or death to personnel;
– damage to or loss of a system, equipment, or property; or
– damage to the environment

Hazards caused or contributed to by software have become a greater concern in systems development as many traditionally hardware-centric systems become highly reliant on software.

Software safety is an ideal example of an emergent product property that cannot be fully tested until the system is operational.
Hazard analysis process

Hazard analysis is a top-down approach to system safety that identifies potential conditions that could lead to loss of life, injury, damage to equipment or the environment.

- **Hazard reports** are created by safety engineers and stored in a hazard tracking system.
- Hazard reports are reviewed at development milestones before development proceeds.
- Hazard analysis is governed by a process document.

Examples:
- Avionics hardware failure results in loss of control.
- The flight computer sends a ‘shut down’ command to the engine control unit during the Ascent phase.
- The engine control unit must verify an ‘emergency condition’ for all ‘shut down’ commands during Ascent.
An example: Software Safety Risk Goals

- **Original question** for a DoD network-centric system of systems:
  Given I have a limited time to test for safety when the system is delivered, what insights can you give me beforehand as to where I should spend my resources?

- **Challenge**: How do we quantify software safety risk to enable proactive assessment and management of software safety?

- **High Level Goals**:
  1. To quantify the importance of software with respect to system safety; and
  2. To quantify the level of risk due to software by leveraging the hazard analysis process
Outline

1. Emergent properties, process risk, and the Process Risk Indicator (PRI) method

2. Example emergent property: Software Safety

3. Applying the PRI to the NASA Constellation Project to measure software safety risk

4. Lessons learned and recommendations
Software safety risk – an emergent property

Case study 1: NASA Constellation program
– a complex system of systems for the next generation of human spaceflight
– We examined three large spaceflight systems in the preliminary design stage
– Each of these systems contained numerous hardware and software safety features.

Case study 2: Unmanned NASA satellite
– Safety-critical software involving propulsion inhibits and solar array deployment
Overview of Constellation Activities

1. Applied our 6-step Process Risk Indicator approach to analyze software safety concerns

2. Examined software safety hazards from the Constellation Hazard Tracking System (HTS)

3. Developed metrics based on software causes of hazardous conditions to provide early insight into software safety
PRI is our 6-step safety process risk approach applicable to emergent system properties, e.g. safety, reliability, …
Step 1: Identify insight areas

- **Inputs**
  - The property you want to measure
  - The processes associated with achieving that property
  - The intermediate outputs of each step for each process

- **Outputs**
  - The set of process outputs or artifacts that should give us the most information about the effectiveness of the process for achieving the property, including:
    - The format of the output
    - Rationale as to how these outputs are of value for identifying the risk of non-conformance or evaluating the effectiveness of the process

- **Sample Activities or Questions to ask**
  - What are the process outputs created during application of the process?
  - How does that information grow or change over time?
  - Can I use this information to gain insight into whether the process is being performed appropriately and if the process is achieving its goals?
Step 1: Identify insight areas

Identify **intermediate outputs** of a process that can provide insights into process conformance and effectiveness

*The set of hazards, with its causes, controls, and verifications*

*The relationship between hazard causes, controls, and verifications over time*
PRI is our 6-step safety process risk approach applicable to emergent system properties, e.g. safety, reliability, …
Step 2: Identify measurement opportunities

• **Inputs**
  – Process outputs/artifacts identified in step one

• **Outputs**
  – Potential metrics based on process outputs/artifacts

• **Sample Activities or Questions to ask**
  – What can I measure to determine if the desired product property (e.g. safety, reliability) is being achieved?
  – What can I measure to evaluate if the process is sufficient for achieving the desired property?
  – Can we identify potential bounds that provide insight for our goals? What is good or bad?
Step 2: Identify measurement opportunities

Evaluate each insight area for information that **could be used to measure** technical or process risks

- Quantify software’s prevalence in hazards by counting hazards, causes and controls with software
- Find the subsystems with the most software risks
- What is the quality of this information?
- Is the information complete and syntactically correct?
Process Risk Indicator Method

1. Identify insight areas
2. Identify measurement opportunities
3. Develop readiness assessment questions
4. Define goals, questions, and metrics for risk areas
5. Develop and enumerate models
6. Propose responses to identified risks

PRI is our 6-step safety process risk approach applicable to emergent system properties, e.g. safety, reliability, …
Step 3: Develop readiness assessment questions

• Inputs
  – Proposed measurement opportunities and the associated risks they measure

• Outputs
  – Advice on how the intermediate outputs and metrics can be used to identify process risk
  – A high-level Indicator of process conformance risk, i.e. are the processes producing meaningful outputs?
  – If step 3 fails, this is a likely indicator of process risk
  – If step 3 succeeds, then continue with the remainder of PRI to measure risk

• Sample Activities or Questions to ask
  – Examine the process artifacts and try to apply the proposed metrics. Can I apply the metric?
    • Is the information accessible and available?
    • Is the information in good enough form that it can be measured?
  – If I cannot apply a metric, why not?
Step 3: Develop Readiness assessment questions

Determine if it is possible to **delve deeper** into the area

e.g., Are the cause, control and verification data available, up to date, and complete enough for analysis?

---

**Example - Avionics failure during ascent results in LoC/LoM**

- **System:** SE+I
- **Element:** SE+I Integrated Analysis
- **Affected System(s):** Ares 1, Orion
- **Subsystem:** SE+I: Avionics
- **Hazardous Condition Description:** TBD
- **Acceptance Rationale:** TBD
- **Likelihood Justification:** TBD

---

Not yet available
1. Identify insight areas

2. Identify measurement opportunities

3. Develop readiness assessment questions

4. Define goals, questions, and metrics for risk areas

5. Develop and enumerate models

6. Propose responses to identified risks

PRI is our 6-step safety process risk approach applicable to emergent system properties, e.g. safety, reliability, ...
Step 4: Define goals, questions and metrics

- **Inputs**
  - A set of proposed metrics that have passed the readiness assessment check

- **Outputs**
  - A GQM structure with specific goals, questions and metrics

- **Activities or Questions to ask**
  - Apply the GQM method to derive a goal template, the questions, and what measures are needed.
    - What is the object of study?
    - What is the specific focus of the measure?
    - What is the purpose of the measure?
    - Who is the person who needs to make a decision about the results of this measure?
    - What are the context variables that might influence the interpretation of the results?
    - Given the goals and questions, what are the metrics?
Step 4: Define goals, questions and metrics

Earlier we defined our goals to be:

- **Goal 1**: Quantify the importance of software with respect to system safety; and
- **Goal 2**: Quantify the level of risk due to software by leveraging the hazard analysis process.
Goal 1 - Prevalence of software

Analyze the available set of hazards in order to characterize them with respect to the prevalence of software in hazards, causes, and controls from the point of view of NASA quality assurance personnel in the context of the Constellation program.

Example Questions

What percentage of the hazards is software-related? A software-related hazard has at least one software cause or software control.

What percentage of hazard causes have software controls?

What percentage of hazard causes are non-software causes (e.g., hardware, operational error, procedural error) with software controls? These causes represent potentially “hidden” software risks.

Example Metrics

The number and percentage of software-related hazards

The number and percentage of software causes

The number and percentage of software controls
Goal 2 - Specificity of software causes

Analyze the **software causes** in a sample set of hazard reports in order to evaluate them with respect to the specificity of those software causes and hazards from the point of view of **NASA quality assurance personnel** in the context of the **Constellation program**.

**Example Questions**

What number and percentage of software causes is **well-specified**, **partially-specified**, or **generically-defined** according to the Constellation hazard analysis methodology requirements?

A well-specified software cause describes all of the following:

- **Origin** – the CSCI (e.g., software component) that fails to perform its operation correctly
- **Erratum** – a description of the erroneous command, command sequence or failed operation of the CSCI
- **Impact** – the effect of the erratum which results in the hazardous condition, and if known, the specific CSCI(s) or hardware subsystem(s) affected

**Example Metrics**

- Count the software causes that are well-specified
### Step 4: Metric values

**GQM for each risk area to expose risks associated with process artifacts**

<table>
<thead>
<tr>
<th>Question</th>
<th>Sys A</th>
<th>Sys B</th>
<th>Sys C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. What percentage of the hazards is software-related?</td>
<td>45%</td>
<td>67%</td>
<td>70%</td>
</tr>
<tr>
<td>2. What percentage of the hazard causes are software causes?</td>
<td>15%</td>
<td>12%</td>
<td>17%</td>
</tr>
<tr>
<td>3. What percentage of hazard causes are hardware causes with software controls (hidden software related hazards)?</td>
<td>14%</td>
<td>11%</td>
<td>-</td>
</tr>
<tr>
<td>4. What percentage of hardware causes has software controls?</td>
<td>16%</td>
<td>12%</td>
<td>-</td>
</tr>
<tr>
<td>5. What percentage of the causes has software controls?</td>
<td>29%</td>
<td>23%</td>
<td>-</td>
</tr>
<tr>
<td>6. What percentage of causes is transferred?</td>
<td>31%</td>
<td>22%</td>
<td>37%</td>
</tr>
<tr>
<td>7. What percentage of controls is transferred?</td>
<td>22%</td>
<td>11%</td>
<td>37%</td>
</tr>
<tr>
<td>8. What percentage of the non-transferred hazard controls are specific software controls?</td>
<td>12%</td>
<td>14%</td>
<td>-</td>
</tr>
<tr>
<td>9. What percentage of the non-transferred hazard controls are references to “generic” software hazards?</td>
<td>5%</td>
<td>2%</td>
<td>-</td>
</tr>
</tbody>
</table>

* System C controls are in a format that prevented accurate assessment of whether the control is software or not.
PRI is our 6-step safety process risk approach applicable to emergent system properties, e.g. safety, reliability, …
Steps 5 and 6

• Step 5: Develop interpretation models and define threshold values
  – For each metric that was measured, define values that represent appropriate process conformance and those values that represent potential risk that the process is not being followed

• Step 6: Propose responses to identified risks, e.g., decisions and actions
  – Propose responses to the early risk identifications that can and should be taken as soon as possible to alleviate the risk
Step 5: Interpreting the data

• **Inputs**
  – A set of goals, questions and metrics to be collected

• **Outputs**
  – A set of models that provides indication that there may be a risk

• **Activities or Questions to ask**
  – Define a set of measures and interpretation models for those metrics, based upon what data is available or can be assumed, to provide indicators that there is a risk that the process is not being followed and the product is at risk of not satisfying the particular property.
    • What is the expected value of that metric and possible margin of error, i.e. what is the range of values that would be acceptable?
    • Do historical data exist for any of the metrics?
    • Are there proxies for the bounds on these metrics?
    • Can we gather any expert opinion on the bounds?
Step 6: Suggesting actions to take

• **Inputs**
  - Metrics and an interpretation model
  - Data from intermediate project artifacts

• **Outputs**
  - Advice on what the project should do if we are outside the acceptable bounds and there is a risk

• **Activities or Questions to ask**
  - Provide expert safety engineer advice on what to do under the circumstances
Steps 5 & 6: Interpreting and responding

Step 5 – develop and enumerate models of how the measures will be interpreted via threshold values.

<table>
<thead>
<tr>
<th>Subsystem</th>
<th>HRs</th>
<th>SW HRs</th>
<th>SW related</th>
<th>% SW HRs</th>
<th>% SW related</th>
</tr>
</thead>
<tbody>
<tr>
<td>Affected subsystem – Top 3 out of 52</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Avionics</td>
<td>26</td>
<td>13</td>
<td>13</td>
<td>50%</td>
<td>50%</td>
</tr>
<tr>
<td>Propulsion</td>
<td>34</td>
<td>12</td>
<td>18</td>
<td>35%</td>
<td>53%</td>
</tr>
<tr>
<td>Command &amp; Data Handling</td>
<td>29</td>
<td>9</td>
<td>14</td>
<td>31%</td>
<td>48%</td>
</tr>
</tbody>
</table>

Step 6 – propose responses to identified risks, e.g., decisions and actions.

1. Allocate additional Software Assurance personnel to design teams and product reviews to evaluate software risk.
2. Require dissimilar command monitoring software on separate partition for all software commands issued to this subsystem.
Some Constellation Highlights

We reviewed the available 154 hazard reports for three spaceflight hardware systems, which included 2013 causes, and 4916 controls.

- ~60% of hazards are **software-related** = software mentioned in cause or control description
- 7% of hazards have “**hidden**” **software risk**
- Identified **traceability risks** in the hazard analysis process and HTS that would impede future verification of controls

Considerable effort was required to analyze and measure software risk, but could be greatly reduced with few changes:

- Additional data fields in the **Hazard Tracking System**
- A **“user guide”** for specifying software causes of hazards
  - disseminated by Goddard Software S&MA
  - Constellation personnel were considering “Letters of Interpretation” to apply the user guide during hazard analysis
Applying PRI to an unmanned satellite

Unmanned satellite

- Safety-critical software involves propulsion inhibits & solar array deployment
- 17% of causes are software causes, 57% of causes software related

As with Constellation, the potential for software-related risk is higher than expected

- Software causes not well-specified
- The cause development guidelines were directly applicable
- Many controls to hardware causes used software solutions. These software controls become potential *causes* of software risk.
- These *hidden software risks* suggest that software safety risk analysis may require more precision
Outline

1. Emergent properties, process risk, and the Process Risk Indicator (PRI) method

2. Example emergent property: Software Safety

3. Applying the PRI to the NASA Constellation Project to measure software safety risk

4. Lessons learned and recommendations
Common software safety process risks

We applied the PRI method to evaluate software safety on three projects to date:

1. NASA Constellation project
2. A large, network-centric US Department of Defense system-of-systems
3. NASA satellite

Three process risks were common across the projects

1. Inability to track software safety hazards and requirements – software safety risks were often not specifically marked in the hazard reports
2. Inadequate traceability – No bi-directional traceability between safety requirements, hazards, causes and controls
3. Inconsistent scope and unstructured details – safety engineers on each project wrote their hazards, causes and controls in unique ways
Software safety institutional challenges

- Integrating software safety with traditional safety processes that originated in hardware and system reliability.

- Defining how software should be incorporated into traditionally hardware-oriented analyses (such as hazard analysis) is still very much a work in progress.

- Elevating software safety to a level of importance equivalent to hardware and system safety was challenging.
Lessons learned for future programs

1. Need to provide explicit guidance for applying safety analyses to software.

2. Need to plan for automated analysis and traceability and promote usage of the hazard tracking system capabilities.

3. Need to require software safety management and measurement in the acquisition process in order for appropriate data to be made available for safety analysis during development.
Summary

• PRI identified metrics based on hazard data to **quantify risk early** in the lifecycle
  – Identified **early risk** in three systems
  – Created a **baseline** for comparison with future review milestones and projects
  – Measures used as input into a **Software Risk Dashboard** to identify subsystems and mission phases with highest software risk

• PRI was able to **identify process problems** and improve the processes
  • Identified where and why the **process was not being followed**
  • Developed **guidelines** developed for safety engineers to describe **software causes** of hazardous conditions
  • Developed a **draft “Handbook”** for incorporating and improving **software assurance oversight for acquisitions** (RFP process)

• PRI demonstrated the benefits of **automated analysis** of early data
  • Built a prototype Hazard Tracking System tool that demonstrated the benefits of additional data and traceability when analyzing software safety risk
Some Publications


Thank you!

Contact information

– basili@cs.umd.edu
– llayman@fc-md.umd.edu*
– mvz@cs.umd.edu