Fraunhofer USA

Center for Experimental
Software Engineering

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

Victor R. Basili
University of Maryland
and
Fraunhofer Center
for Experimental Software Engineering, Maryland

ISSRE 2008

Outline

- Problem
- Approach
- Safety Application
  - Development
  - Independent Test
- Benefits
- Future Work

The Problem as Stated

- The problem posed was how can we aid the software safety assurance engineer
  - Monitor the quality of the software safety process
  - Identify areas of risk where software safety is not being appropriately addressed
  - Provide support for mitigating these risks
  - Be able to recommend potential solutions for improving the software development

- Could we propose as set of measures that can be
  - Integrated into the general measurement program
  - Cost effective by leveraging data already collected

Context

- The number of complex system of systems has grown
  - Bigger and more automated
  - Distributed and network-connected, not stand alone
  - Composed of hardware and software elements
  - Uses many suppliers
  - Structured in many layers of suppliers
- All subsystems need to be integrated and use different approaches for dealing with safety
- Software has a much larger role in the system than ever before, e.g., the networking

The Problem in the Abstract

- We can define functional and non-functional requirements for a system and there are processes, methods, and techniques we can use to achieve these requirements with respect to the system, e.g. on-time delivery, cost, quality

- Projects monitor development to see if they are achieving these requirements by studying the current amount of the resource itself , e.g., time expended, and comparing it against some other some other resource, e.g., amount of product created
  - e.g., for on-time delivery they might check the % of code, classes, or components developed according to some pre-defined schedule at various milestones

The Problem in the Abstract

- What happens if we cannot measure a resource expenditure? What do we do if we cannot evaluate the requirement until the system is completed?
- For example, suppose we are considering trust requirements, such as safety, security, reliability. What resources are these related to?
- If we cannot measure if a characteristic exists early on, can we at least measure if there is a risk that the characteristic ***will not exist***
- How can we recognize the ***potential risk*** during the development of a system of not achieving the desired trust characteristic?
- How can we do this with minimum cost and overhead?
- Note, we may not be able to show the trust characteristic is there but we can show it cannot be there

One Approach

- **Assumption**: There is a relationship between the processes used during software development and the product's characteristics
  - **Characteristic (Product) = F (Process, People, …)**

- **Implication**: A lack of process suggests there is a risk of not achieving the proper product characteristics

- **Opportunity**: If we analyze the execution of the process, we can provide insight into whether appropriate processes are being performed

- **Approach Idea**: Analysis of intermediate outputs during development can help provide insight and help identify potential risks in achieving the desired characteristics

Visibility Into System Risk

- **Goal** is to develop and implement a set of measures that provide management ***visibility*** into the system (and software)

- For the **purpose** of asking the right questions, identifying risks and monitoring the quality of the trust-related process

- What is happening?
  - What kind of information can be gathered from development that will make the lack of process visible and provide insights into the focus, amount, and types of analysis needed?
  - What are the high risk issues for this system with respect to the anticipated characteristic?

Visibility Approach

- Measure process *OUTPUTS*, intermediate products generated during development and test

- **Is sufficient material there?** Where are the potential risks based upon missing information?
  - This is a syntactic, quantitative analysis.
  - Can be measured directly; can be automated

- **Is the right material there?**
  - This is a semantic analysis
  - Can generate statistical samples, based upon the lack of sufficient materials, that can be manually inspected for quality attributes, e.g., correctness

Visibility Approach

- Apply a set of models and measures to objectively assist in identifying risk areas where the characteristic may not have been properly addressed

- Use available information to focus the analysis
  - Understand what data is available and how we might interpret that data
  - Whenever possible, use existing data (don’t impose additional costs, time burden)

- As an example, consider the trust requirement of safety with the fundamental question:
  - **Is safety being dealt with appropriately?**

Some Software Safety Terminology

- **Hazard**: Any real or potential condition that can cause injury, illness, or death to personnel; damage to a system, equipment, property, or the environment

- **Hazards** recorded in a Hazard Tracking System **HTS** have the format:

  (Identifier, Scenario (Description), {Causes, {Control, {Verification}}})

- **Trouble Reports**: Are recorded in failure/trouble-reporting system and are assumed to be marked as safety-related

- **Properties of software hazards**:
  - A hazard is a *software hazard* if it has at least one software cause
  - A *hazard* is *open* if at least one of its causes is open
  - A *cause* is *open* if at least one of its controls is open
  - Hazards are retired (risk is reduced) but never leave the data base
  - *Closed* hazards can become *open* hazards when a new cause is found

Defining Measures to Provide Insights into Software Safety

1. **Identify** potential **insight areas** and intermediate outputs that cover the important aspects of the safety process for the specific environment
2. **State** the **purpose** of each insight area serves
3. **Ask** a set of **Readiness Assessment** questions that
   - Provide initial insight into the areas of interest
   - Allow a quick and easy status report of the area
   - Identify whether it is possible to go deeper into the area
4. **Define Software Safety Visibility Goals and Questions** to expose risks associated with outputs of the safety process
5. **Develop Measures and Models** to define what will be measured and how it will be interpreted
6. **Identify Responses** to potential risks indicated by measures outside the model thresholds and further actions to be taken
7. **Apply** the measures and interpret the results

1. Identify Insight Areas

1. Identify potential **insight areas** and intermediate products that sufficiently cover the important aspects of the process for the specific environment.

- What is the level of detail of the process being applied to develop the system with the safety requirement?
- What are the intermediate outputs of the process?
- What are the potential insights that can be gained from those outputs?

- Considerations for selecting areas depend on information and data available, processes/ technologies used, life cycle being followed, historical data pointing to specific problem types, contribution to insights, cost and schedule, ..

1. Identify Insight Areas

- Based upon the considerations, we selected the following **insight areas** (and intermediate outputs) to support program management and the safety engineer

  - **Software Safety Analysis Process** (Process Document, Requirements Document,)
  - **Hazard and Mitigation Identification (HTS)**
  - Hazard Monitoring (HTS)
  - Appropriate Level of Rigor for Software Safety (Process Document)
  - Safety Defects (TR)

Example: **Hazard and Mitigation Identification**

Look at the Hazard Tracking System to see if information is put in it

2. State Purpose

2. State the **goals** associated with each insight area (what is the insight that the output provides)

- What are the goals for examining those intermediate products relative to the insights they provide?

- Example: **Software Safety Analysis Process**
  - Confirm that system and software requirements and development practices are in compliance with safety processes

- Example: **Hazard and Mitigation Identification**
  - Ensure that the program is adequately executing the safety process by identifying and documenting the appropriate information

3. Ask Readiness Assessment Questions

- The purpose of the **Readiness Assessment** questions is pragmatic

- Before delving into the effort of the developing models and measures and collecting data we propose a set of questions that allow us to
  - Gain some initial insight into the areas of interest
  - Get a quick and easy status report of the area
  - Identify whether it is possible to go deeper into the area

- It is possible that the some process aspect is so poorly applied or misunderstood and the project needs to correct this as early as possible

- What is learned from these questions helps tailor the models, measures, and responses applied at the deeper level

3. Ask Readiness Assessment Questions

- Example: **Software Safety Analysis Process**

Is there a documented software safety process that identifies requirements as safety-related?

Are safety-related requirements marked as such in the requirements repository?

4. Software Visibility Goals and Questions

- We now know the artifacts exist and can be analyzed.

- We are at the level of setting goals for what we expect to see from the artifact we are examining

- Example: **Software Safety Analysis Process**

Goal: Make each sub-contractor and the integrator safety processes visible by checking on the items it has identified

Question: Are there a reasonable number of software safety-related requirements being identified?

5. Develop Measures and Models

- Here we use a measurement template to provide the best interpretation possible depending on what other data or expertise is available
- We take the **question** posed and define a metric that captures the concept addressed by the question
- The **metric** can be any basic measure or derived metric, as well as evidence of some sort (e.g., existence of documents or processes)
- We then select a **model** that defines the expected mathematical bounds on the metric and the **interpretation** of the values of the metric(s), in order to answer the target question

5. Develop Measures and Models

- For each model we make assumptions about how the metric values should be interpreted. This involves the selection of an **expected value** and a **range** for that expected value.

- We use several approaches for estimating the **expected value**:
  - Historical data from past projects such as an average value of data collected from past projects that are similar to this project
  - Prior data from the current project such as the average for all platforms on this project within the family that should have similar responses.
  - Proxy Estimate if the expected values of the current model behave like some other variable or equations that can be measured.
  - Expert estimate by using an estimated value selected by an expert or group of experts

5. Develop Measures and Models

- The **range** of the expected value is one of the following:
  - Based upon a normal or other distribution such as some function of the standard deviation from the mean.
  - Based upon the distribution determined by the values used to make the proxy estimate such as a running average over several points on a curve.
  - An expert estimate of the range.

- In general, if the **calculated value** is
  - Less than the Estimate ± range, then there may be a development problem
  - Greater than the Estimate ± range then we may have planned wrong and need to reconsider cost and schedule

- The estimate of any expected value or range should be improved over time based upon new information.

5. Develop Measures and Models

- To this collection we add a **scope** of application of the metric, e.g., we can assume these metrics are taken for certain suppliers or certain types of systems

- And a suggested **response**s to the application of the model being within or without bounds

- So the measurement **template** consists of:
  - **Question** being addressed with the aid of the metric
  - The **metric** definition
  - The **model/interpretation** recommended
  - The **scope** of application of the metric
  - Suggested **response**s to the application of the model

5. Develop Models and Measures

Question: Are there a reasonable number of software safety-related requirements being identified?

Measure: **PSSR =** # software safety requirements / # software requirements

Model:

***if* |PSSR - EPSSR| < e**

where

EPSSR is the estimated value of PSSR,

e is the acceptable threshold for deviation from the estimate (EPSSR – e, EPSSR +e) is the acceptable range,

then a reasonable number of software safety requirements have been identified

5. Develop Models and Measures

Question: Are there a reasonable number of software safety-related requirements being identified?

Calculating the value and range of EPSSR

If we have historical data for similar systems, we can let

EPSSR = the average of the PSSRs for all similar systems

e = σ(EPSSR)

or

We can define a proxy, such as assuming the relationship is in line with system safety in general, then

EPSSR = #system safety requirements / #system requirements , and guess at e based upon expert opinion, e.g.,

e = 20% of EPSSR

6. Identify Responses

Question: Are there a reasonable number of software safety-related requirements being identified?

Measure: **PSSR =** # software safety requirements / # software requirements

Response:

If PSSR is not within the range of EPSSR

then there is a need for a management action,

check if the safety analysis process is being applied right;

come up with a “get well” plan or investigate the reason why the system under consideration has such a small (or large) number of software safety requirements

If too large

then what are the cost and schedule implications?

Approach

1. Identify **Insight Areas** and intermediate outputs
2. State the **Purpose**
3. Ask the **Readiness Assessment Questions**
4. Define **Software Safety Visibility Goals and Questions**
5. Develop **Measures and Models**
6. Identify **responses**
7. **Apply**

Example Steps and Measures

- We have applied this approach to the development of a DoD safety critical complex system of systems
  - It provided insights into problems during development to program management
  - It was effective in pointing out a number of risk areas that were not getting sufficient attention

Sample Problems Identified

- Problem identified with various suppliers:
  - Software-related hazards not marked as such
  - Hazard controls not identified as software related
  -
  - Hazards not fully traceable to the source of the hazard
  - Hazard controls not traced to the requirement specifications.

  .
  - Verification data missing from in the HTS.

Sample Responses

- **Reanalyze and update** the data in the HTS to
  - correctly identify hazards and causes as software related

- **Add guidance to the HTS user guide** to
  - more clearly define a software hazard, i.e., a hazard is a software hazard "if it has at least one software cause or one software control" and communicate this guidance to the users

- **Modify functionality of the HTS** to
  - allow tagging controls as software related appropriately.
  - allow bi-directional tagging of controls to the requirements tool

The Second Problem

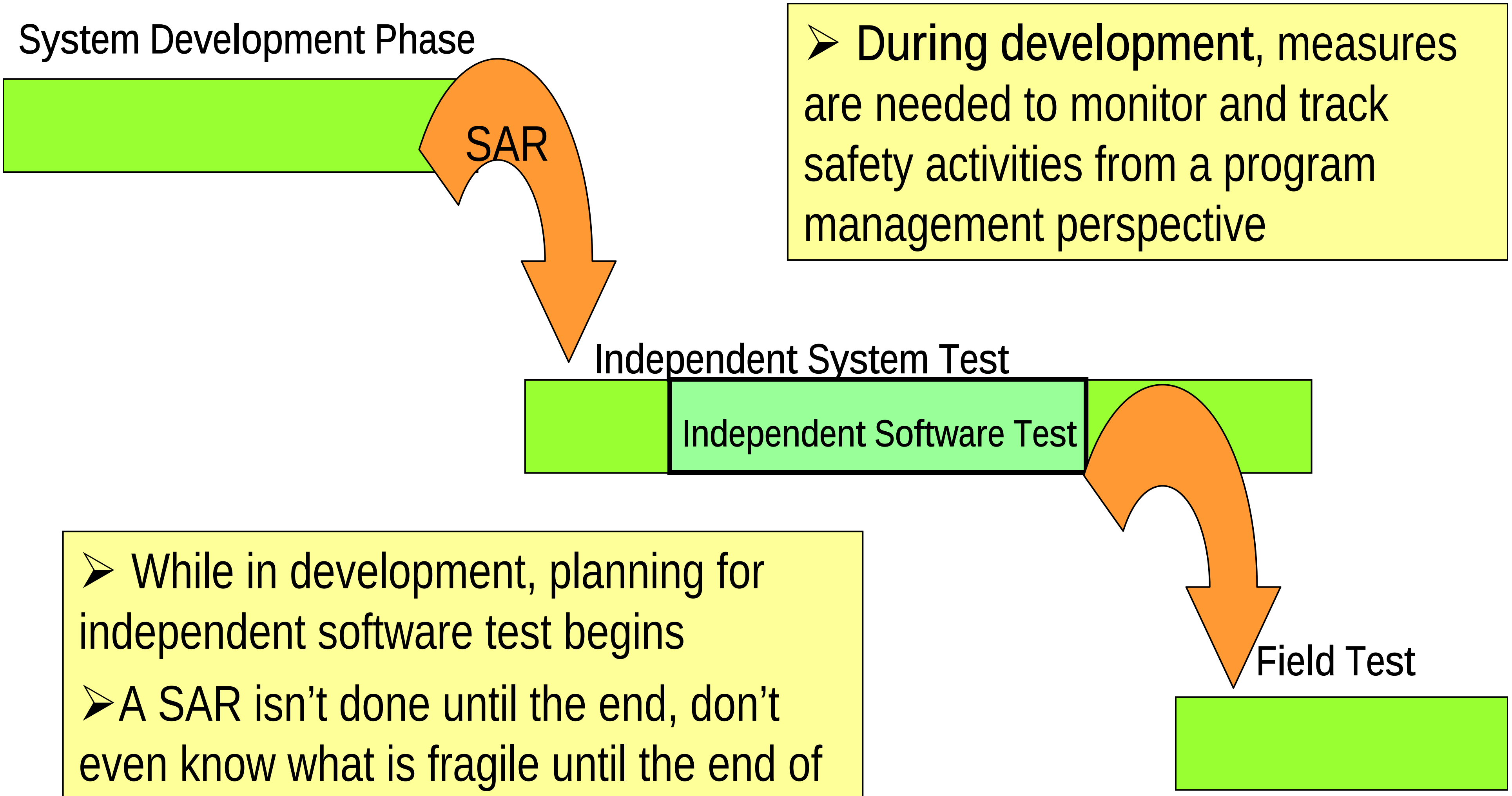
- There is a need to improve the safety analysis during **independent software test** to gain more confidence in the safety of a system

- **Independent safety evaluation** of a system is traditionally done at the end of the system’s development life cycle

  →Late visibility into problems, limited time to do analysis and test

- How do they *maximize the opportunity* of identifying potential safety risks during independent test?
- How do we *take advantage* of risk identification information from development in a cost effective way?
- How do we *focus and evaluate* safety activities during independent safety test?

Context



System Development Phase

SAR

Independent System Test

Independent Software Test

Field Test

- ➢ **During development**, measures are needed to monitor and track safety activities from a program management perspective

- ➢ While in development, planning for independent software test begins
- ➢A SAR isn't done until the end, don't even know what is fragile until the end of the development phase

The Expanded Process Steps for Independent Software Safety Test

A. Apply the approach during **development** and this information is available to independent software test for planning purposes

- Provides program management with visibility into development

B. This data can be used for **planning independent software test**, by creating new goals, measures, models, or responses

- Apply a modified approach, constrained by available data
- Permits planning a more efficient independent test

C. Apply the approach to the **execution of independent software test** phase, identifying new areas of interest, goals, metrics, models, etc.

- Increases confidence in the safety of the released system

B. Software Safety Risk Reduction for Independent Software Test Planning

1. **Insight Areas:** focused for independent software test planning
2. **Insight Area Goals:** may be same areas used during development phase, but looked at them more from an independent safety test/analysis perspective
3. **Readiness Questions:** Do we have sufficient data from development to support each of these new goals?
4. **Software Safety Visibility Goal/Questions:** can very within limits
5. **Measures and models:** can very within limits
6. **Responses:** modified to focus on independent test actions
7. **Apply**

B6. Identify Responses

Development Response:

If PSSR is not within the range of EPSSR
then there is a need for a management action,
check the safety analysis process and whether it is being applied right;
come up with a "get well" plan …

Independent Test Response:

if PSSR is not within the range of EPSSR **then** :

if too small

then if safety requirements are not identified **then**

developer test cases may not be sufficiently robust

assess if requirements were correctly allocated to safety

explore developing more robust test cases in Independent Test

If too large

then likely independent test will have to be more comprehensive; longer duration/more robust.

C. Software Safety
Risk Reduction For Deployment

1. Identify **insight areas** that cover the independent test activities
2. Focus the **goals** associated with each insight area on the evolving product in independent test
3. Apply a set of **Readiness Assessment** questions that
   - What data do I have from development to jump start my analysis, e.g., estimated bounds and ranges?
4. Define/focus **Software Safety Visibility** goals and questions to expose risks associated with outputs of the safety analysis process
5. Develop/enumerate **measures and models**
6. Identify **responses** to potential risks indicated by measures outside the model thresholds and further actions to be taken
7. Apply the measures and interpret the results

Example insight areas and questions for independent software test

Potential insight areas that support development and tailoring of independent safety test

1) Review of Hazard Tracking System (HTS) Data
2) Analysis of Software Requirements
3) Analysis of Software Design
4) Review of Contractor Software Problem Reports (SPRs)
5) Analysis of Developer Software Test Planning and Execution
6) Review of Safety Assessment Report (SAR)

NASA Safety Metrics Project

- NASA has a set of robust software safety process guidelines applicable across projects
- We are working with the Constellation Program to
  - Create a set of measures tailored to those guidelines
  - Save these measures an “Experience Base” that will
    - support decision making by the software safety engineer
    - provide visibility into the software safety process for multiple stakeholders
    - identify implementation practices (good and not as good) for implementing the guidelines.

ISSUES

It is a critical assumption that there is a direct relationship between process and product

Forces you to state up front what that relationship is

Forces you to predict what should happen

A lot can be learned if you are right (or wrong) in terms of building knowledge – empirical study

Are we applying measureable processes for other trust characteristics, e.g., safety, reliability, privacy, …

Benefits

- Using the relationship between process and product has many advantages
- It creates early visibility into potential risks and provides management with insights
- Is a low cost, high benefit approach
- Has been applied for safety successfully
- Offers an evaluation of the safety activities for the safety engineer
  - Increases confidence in the safety of the released system
  - Identifies risks resulting from the application of the safety hazard analysis process (or lack there of) and assesses the *potential* for achieving a safe system

Metrics will not tell us whether the system is safe, but they provide indicators of potential problems and risks.

Future Work

- How do we expand this concept for other trust characteristics?
- The more we know about the process, the more we can identify risks
    - Does this make sense for safety, reliability
- It assumes we know something about the relationship between process and product
- How do we incorporate this into the normal acquisition and development processes?
- We have begun to look at insight areas for ordinary project management

The Team

- Victor Basili, University of Maryland & Fraunhofer Center – Maryland
- Frank Marotta, *U. S. Army Aberdeen Test Center*
- Kathleen Dangle, Fraunhofer Center – Maryland
- Linda Esker, Fraunhofer Center - Maryland