Fraunhofer USA

Center for Experimental Software Engineering

A Methodology for Using Measures to Assess Software Safety Risk from an Independent Testing Perspective

Victor R. Basili, Kathleen Dangle, Linda Esker

Fraunhofer Center
for Experimental Software Engineering, Maryland

ITEA Symposium

Outline

The Problem

- Independent evaluation of the safety of a system is traditionally done at the end of the system's development life cycle, i.e., during independent test, (e.g., DT)
  - Late visibility into problems
  - Limited time to do analysis and test
- Resources during independent software test for safety are limited
  - Time and effort are limited resources
  - Need to be used effectively
- There is a need to improve the safety analysis during independent software test to gain more confidence in the safety of a system
- There is a need to maximize the opportunity of identifying potential safety risks that may not be exposed during operation

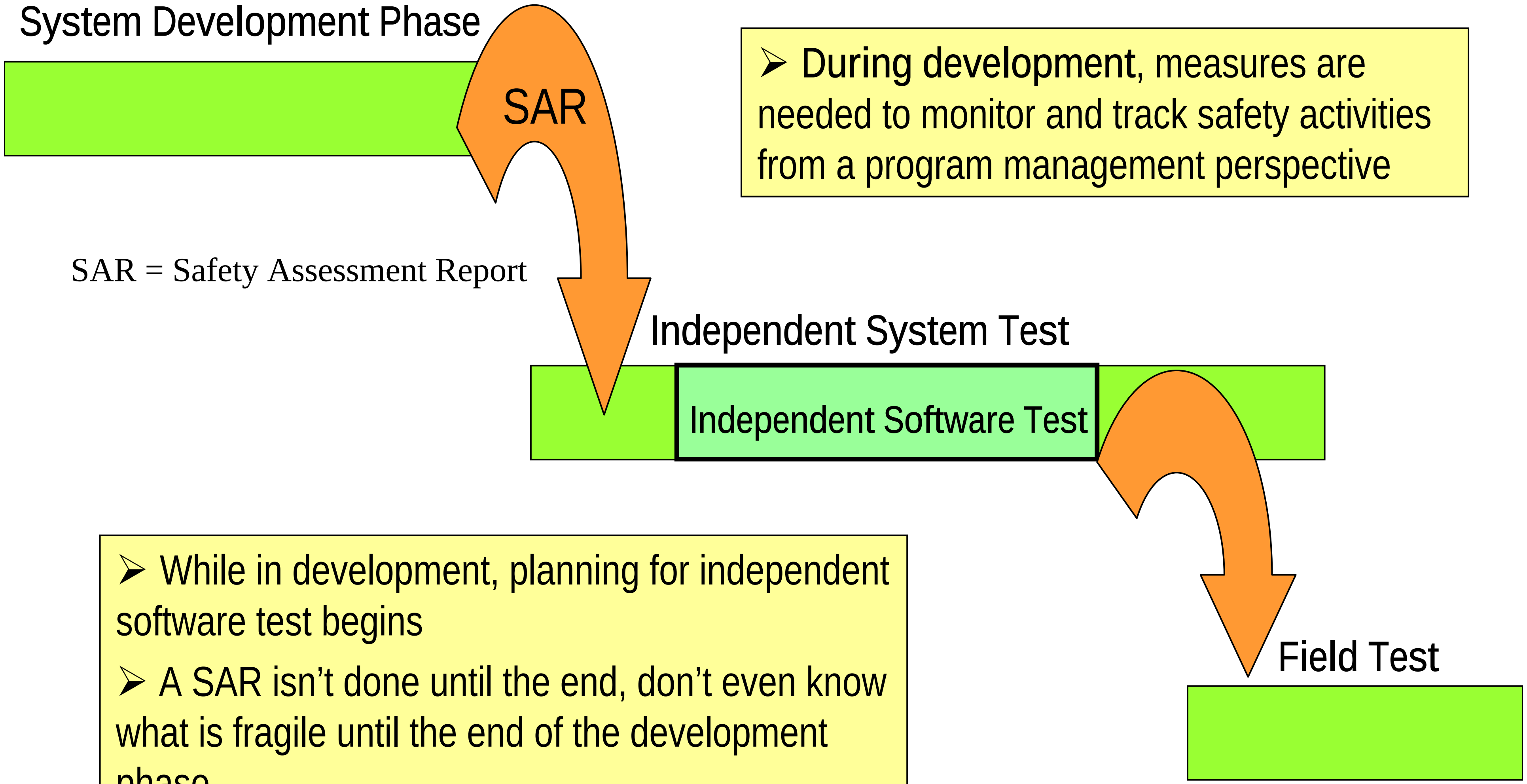
The Problem: More specifically

- There is insufficient start up information to assess the cost and schedule for independent testing:
  - How do we *plan* effective use of resources?
- Developer processes are insufficient or lack safety deliverables
  - What kind of useful information can we *gather?*
  - How do we do it *contractually*?
- There is a need to focus resources by understanding where the higher risks are
  - How do we *take advantage* of this information in a cost effective way?
- There is a need for assessment of independent software safety test
  - How do we *focus and evaluate* our activities?

Fraunhofer USA

Center for Experimental
Software Engineering

Context



SAR = Safety Assessment Report

- **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

Fraunhofer USA

Center for Experimental Software Engineering

Context

- From a safety point of view, in **independent software test**
  - Testing and analysis are both required
    - Analysis involves assessing the hazards, the causes, the controls, and the verification for completeness and correctness, and testing involves checking that verifications on controls are complete, regression testing of those verifications as new changes are made, etc.
  - Testing and analysis are complex
    - Emphasis on “rainy day” testing vs. “sunny day”
    - Software by its nature introduces more off-nominal and out-of-bounds cases into the system
  - It is the last milestone focused on assuring software safety

Visibility Into System Safety Risks

- What happened before? What is independent software test receiving?
  - What kind of information can be gathered from development that will provide the testers insights into the focus, amount, and types of analysis and testing needed?
- How can we leverage prior safety activities performed, so that independent software testing can be tailored to the system it is receiving?
  - What functionality of the system is ready for independent software testing?
  - What are the high risk safety issues for this system?
- How can we measure our progress during independent test and prioritize our activities according to highest safety risk issues?

Approach

- **Goal** is to develop and implement a set of metrics that provide management ***visibility*** into system (and software) safety
- For the **purpose** of asking the right questions, identifying safety risks and monitoring the quality of the safety process
- 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

Fraunhofer USA

Center for Experimental Software Engineering

Approach

- Apply a set of metrics to objectively assist in identifying areas where safety may not have been properly addressed

- Use development knowledge to focus analysis and test
  - Understand what data is available and how we might reinterpret that data from an independent test viewpoint
  - Develop an independent test plan that focuses on high risk areas
  - Whenever possible, use existing data (i.e., do not impose additional costs, time burden)

- For example
  - During development perform this syntactic and semantic analysis
  - Make data available to independent software safety tester for planning
  - During independent test, perform this syntactic and semantic analysis to provide insight into safety concerns

Defining Measures to Provide Insights into Software Safety

1. Articulate the purpose of the safety related activity and Identify potential **insight areas** that sufficiently cover the important aspects of the software safety process for the specific environment
2. State the **goals** associated with each insight area
3. Develop 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 analysis process
5. Develop/enumerate **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

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

- To illustrate the approach and the kinds of measures and models that can be used, we use a sample from those goals, questions, measures and models to demonstrate the specifics of the process

- We then extrapolate to the activities relevant to independent software test

1. Identify Potential Insight Areas

- Considerations for selecting areas may depend on
  - Information and data available
  - Processes/ technologies used
  - Life cycle being followed
  - Historical data pointing to specific problem types
  - Contribution to insights
  - …
- Example **program management** insight areas
  - Software Safety Analysis Process
  - Hazard and Mitigation Identification
  - Hazard Monitoring
  - Appropriate Level of Rigor for Software Safety
  - **Safety Defects**

Select areas based on cost and schedule constraints

Safety Defects: Steps 2, 3, and 4

2. State the **goals** associated with each insight area

   Insight Area Goal: Identify whether any safety problems remain in the system for the Safety Assessment Reports (SARs) by verifying that all safety controls/ requirements have been tested

3. Develop a set of **Readiness Assessment** questions
   - Are safety-related failures/faults identified as such in the Software Problem Reporting System?
   - Are safety-related test cases identified as such?
   - Are defect closures recorded?

4. Define **Software Safety Visibility** goals and questions to expose risks associated with outputs of the safety analysis process

   Goal: Check if software safety-related defects are being dealt with appropriately

   Question: Are software safety-related defects being closed at a reasonable rate over time?

Safety Defects: Steps 5 and 6

5. Develop/enumerate **measures and models** to define what will be measured and how it will be interpreted

   Measure: **COSRTR** = count by priority of open safety-related software trouble reports at time i

   Model: If **COSRTR** ≠ 0 then there are open defects that need further analysis

6. Identify **responses** to potential risks indicated by measures outside the model thresholds and further actions to be taken

   Development Response: If all safety related defects are not closed, then create the list of open defects, prioritize and investigate the reasons. This measure should be taken periodically starting at the beginning of test, up until SAR delivery

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

Fraunhofer USA

Center for Experimental Software Engineering

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**

Fraunhofer USA
Center for Experimental Software Engineering

Safety Defects: Steps 5 and 6

Measure: **COSRTR** = count of open safety-related software trouble reports

Model: If **COSRTR** ≠ 0 then there are open defects that need further analysis

Development Response: If all safety related defects are not closed, then create the list of open defects, prioritize and investigate the reasons. This measure should be taken periodically starting at the beginning of test, up until SAR delivery

Independent Test Response: Given the list of safety related defects not closed:

(1) Assess their impacts on safety and determine in coordination with safety community which problems are 'must fix' for immediate use or can be deferred.

(2) Plan for robust independent test, including them in the sample set of issues to be semantically checked.

Fraunhofer USA

Center for Experimental
Software Engineering

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)

Fraunhofer USA

Center for Experimental Software Engineering

Benefits

- Leverages development activities/data to plan independent software test
  - Helps make efficient use of test resources
- Makes clear what is needed for the safety engineer to make maximum use of independent test resources for safety
- Provides for independent test activities to focus on high risks in a cost effective way
- 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.

Fraunhofer USA

Center for Experimental Software Engineering

Future Work

- The expanded processes for independent software test is preliminary and has not been applied
  - We would like to identify potential systems for application
- The remaining question: How do we incorporate this into the whole acquisition process?
- We believe the essence of the approach can be applied to independent test in general, not just for safety

Fraunhofer USA

Center for Experimental Software Engineering

Contact Information

Vic Basili
Fraunhofer Center Maryland
University of Maryland
basili@fc-md.umd.edu
301-403-2705

Kathleen Dangle
Fraunhofer Center Maryland
kdangle@fc-md.umd.edu
301-403-8973

Linda Esker
Fraunhofer Center Maryland
lesker@fc-md.umd.edu
301-403-8967