Using SysML to Support Software FMEA

Phillip Schmidt
The Aerospace Corporation

2012 Flight Software Workshop
November 2012, San Antonio, Texas

PUBLIC RELEASE IS AUTHORIZED.
Agenda

• Motivation
• Prior Work
• Approach
  – Requirement Discovery
  – Integrating Use Cases and SW FMEA
  – Applying SysML to Specification Conformance
• Future Work
Motivation

• Recent NASA Fault Management workshops indicate that application of SW FMEA to flight software has been limited
  – *FMEA generally applied to HW components*
  – *Little formalization in tying FMEAs with software architectural support*
  – *Dependency analysts and software developers use different technologies*

• Understanding the effects of software resources and design choices to mitigate certain off-nominal conditions is essential.
  – *Message corruption*
  – *Queue overflow*
  – *Device malfunction on software behavior*

• Prior research suggests that SysML and UML can help close the gap between design development and dependability analysis

• Some techniques and approaches are presented
Some Prior Work

• Leangsuken, et. al 2003
  – Assuming that dysfunctional behavior is known, capture it using UML models
  – Defined new UML stereotypes to describe dysfunctional behavior
  – Requires high quality UML representations to enable analysis

• Guiochet, et. al 2004
  – Applied UML in medical robotics domain to do risk analysis
  – Assessed functional behavior only
Prior Work (cont’d)

• David, et. al 2008
  – Goal was to close gap between early conceptual design engineers and dependability analysts
  – Found that dependability estimation was not always well mastered
  – Wanted to automatically deduce dysfunctional behavior from conceptual design model (UML)
  – Analyzed sequence diagrams of conceptual UML models
    • Auto-generated FMEA entries for every message sender (e.g. cause) /receiver (e.g. effect)
    • Failure modes characterized as partial function, no function, intermittent function, unintended function
    • Generated too many inputs that needed user intervention
  – Later work for automated FMEA synthesis used a SysML model to integrate a component database to characterize the failure modes
Prior Work Lessons

- Using UML/SysML for reliability analysis is appropriate
- Prior work has focused on functional component behavior
  - Can lead to simplified view of present/absent component behavior vs degradable service provided by software design
- Automated FMEA synthesis without model context understanding can generate large amounts of data requiring manual intervention
- A dysfunctional dictionary of component failure information can help focus component failure analysis
- Work on addressing software design choices is still needed
Challenges

• High quality UML/SysML models are not always available
  – In agile-base development requirements and design evolve throughout the lifecycle
  – Design-code alignment studies of flight software show that faithful UML representations may only capture 40% to 60% of the implementation
  – Basing automated analysis on incomplete information could be misleading

• FMEA knowledge has generally been HW component based
  – Component FMEA data not discussed in terms of use case scenario

• Promote software design that recognizes design choices that can impact behavior and provides effective off-nominal options when needed
Approach

• Realize UML models (and SW FMEA insights) evolve over lifecycle
• Approach application of SysML as part of risk mitigation effort
  – Perform requirement, concern, and goal discovery using classic SysML
  – Consider integration of use cases and SW FMEA
  – Leverage SysML framework to conduct specification conformance (e.g. functional failure analysis)
• Extend UML model with SysML model annotations
  – Use stereotype extensions to characterize expected requirements, derived constraints, off-nominal conditions for test and their status
  – Apply stereotype annotations as part of peer-review discovery, legacy FMEA insight
  – Define figures of merit (assessment indicators) to evaluate constraint compliance, specification/test, FMEA conformance and completeness
  – Even recommendations not accepted can be evaluated in a context
• Develop/use automated tools to perform analysis
  – Not all data needs to be embedded in SysML
  – Back end analysis can be tied into SysML parametric models
Requirement Discovery Management

Reqt Spec

import

Reqt

<<refines>>

Reqt

<<discovered>>

Reqt

<<discovered>>

Discovered Off-nominal Reqt

Other associations:
Class Behavior
Test Cases
Use Cases
Activity, Sequence, State Diagrams

Verification
Figure of Merit

Constraint condition evaluation
Constraint condition evaluation
Constraint condition evaluation

Parametric Model

Other pre-condition triggers

Identify:
SW FMEA failure modes,
Mitigation
Design option rationale
Recommended tests
Integrating Use Cases and SW FMEA

• Use cases and SW FMEA generally viewed separately
  – *Use cases describe functional behavioral requirements*— as nominal, alternate, exception/error courses of action
  – *SW FMEA describes failure modes of software resources and mitigation of those failures*
  – *Scope of SW FMEA is broader than functional requirements as design choices may affect operations, non-functional behavior*

• Advantages:
  – *Integration provides problem context and would clarify its mitigation approaches*
  – *Provides a functional framework in which classes of failure modes could be addressed* (e.g. failure of pre-condition/post-conditions are failure mode detections)

• Challenges:
  – *Step granularity and crosscutting concerns affect scalability, manageability*
  – *Automated support needed for efficiency*
Use case-SW FMEA Integration

• Approach:
  – *Formalize use case steps via parameterization of pre-conditions, post-conditions and actions*
  – *Extend the use case schema to identify failure modes and their mitigation*
  – *Incorporate Requirement Discovery approach*
  – *Modularize use case processes (e.g. log function checks)*
  – *Autocorrelate legacy FMEA database with use case data*
  – *Operationalize non-functional requirements via benchmark tests*
  – *Include design assumptions*
  – *Link to other UML/SysML artifact references to manage granularity*
    • E.g. Specification references, UML/SysML diagrams, specialized stereotypes such as goals
  – *Develop automated analysis tools to check for consistency*
Use Case – SW FMEA integration

- Example uses a table representation here for conciseness.

<table>
<thead>
<tr>
<th>Req ID</th>
<th>UC Module Name</th>
<th>UC Step</th>
<th>PreConditions</th>
<th>Failure Mode Detection</th>
<th>Description</th>
<th>Action</th>
<th>Event</th>
<th>Nominal Post-Conditions</th>
<th>Requirement Type</th>
<th>Parent Req</th>
<th>Verification</th>
<th>Resources</th>
<th>Comments</th>
<th>SW FMEA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Req_ATT-10</td>
<td>AttitudeRequest</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Req_ATT-10.1</td>
<td>AttitudeRequest</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Req_ATT-10.2</td>
<td>AttitudeRequest</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Req_LOG-10</td>
<td>LogService Req</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Req_LOG-20</td>
<td>LogService Req</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Not all discovered requirements may survive in agile development. This mitigation affects operations. Using a message queue design could obviate this.**

**These requirements capture off-nominal behavior and mitigation.**

**Resources and diagram links assist correlation with legacy FMEA data.**

**Recommended test.**

**Discovered requirements not currently in baseline.**

**This impact could force a redesign of Log Service.**
Applying SysML to Specification Conformance
Future Work

• As models evolve want to develop automated ways to apply model annotations
  – Requires understanding model differences over time
  – UML/SysML supports model transformations
  – Looking at applying pattern analysis using Prolog

• Strong desire to assess design choice impacts on mission accomplishment
  – Formulate mission accomplishment levels with reward-based models using performability analysis
  – Assists in assessing different fault management strategy/goal options
    • Adaptive goal-oriented fault tolerant systems
    • Graceful degradation
    • Multiple fault design robustness
References


Backup
Fault Management Bus Reference Model
Requirements Constraint Management

- **Functional Requirement**
  - **Id**: "124"
  - **Risk**: Low
  - **Source**: ""
  - **Text**: "(U) When a failure check is enabled, the $FSW shall detect a failure when the conditions defined by the failure check are satisfied."
  - **Verify Method**: Test

- **Functional Requirement**
  - **Id**: "117"
  - **Risk**: Low
  - **Source**: ""
  - **Text**: "(U) When a failure check is enabled, the $FSW shall provide the capability to perform a failure verification."
  - **Verify Method**: Test

- **Functional Requirement**
  - **Id**: "52"
  - **Risk**: Low
  - **Source**: ""
  - **Text**: "(U) When the $VS is Ascent, Normal Operations, Autonomous, or $MODE_SAFED and the $SOTC is enabled, the $FSW shall detect, isolate, and verify a thruster which is stuck open as specified in the $FFA."
  - **Verify Method**: Test

- **Functional Requirement**
  - **Id**: "56"
  - **Risk**: Low
  - **Source**: ""
  - **Text**: "(U) When the $VS is Ascent, Normal Operations, Autonomous, or $MODE_SAFED and the $SOTC is enabled, the $FSW shall detect, isolate, and verify a thruster which is stuck open as specified in the $FFA."
  - **Verify Method**: Test
Fault Management Isolation Map
Constraint Evaluation