

Software Engineering for Safety: A Roadmap

Robyn Lutz
Jet Propulsion Laboratory/CIT
rlutz@cs.iastate.edu
<http://www.cs.iastate.edu/~rlutz>
ICSE 2000
June 9, 2000

**Funding provided under NASA's Code Q Software Program Center
Initiative UPN #323-08**

Current State

- **Increasing reliance on software in safety-critical systems (SCS)**
 - **Processing improvements**
 - **Miniaturization**
 - **Virtual environments**
 - **Market forces**
- **Software can contribute to a system's safety**
- **Software can compromise a system's safety**
- **Where do we stand in building and understanding software in SCS? (Where do we fall?)**

RRL-FSE/ICSE00

2

Roadmap

- **Talk describes seven directions for needed work in software engineering for safety**
 - **Challenge:** perceived distance between current state & the goal
 - **Approach:** proposed path to reduce distance
- **Criteria:**
 - **Important for actually building safer systems**
 - **Approaches to solutions are indicated in the literature**
 - **Significant progress appears feasible in next decade**

RRL-FSE/ICSE00

3

Directions

1. Further integration of informal and formal methods

- **Challenge: Provide readier access to formal methods for developers of SCS**
- **Approaches:**
 - **Automatic translation of informal notations (developers' descriptions) into formal notations (to support analysis)**
 - **e.g. graphical and tabular representations, fault trees, UML, visual programming environments**
 - **e.g., model checking, theorem proving, executable specifications, rigorous reasoning**

RRL-FSE/ICSE00

4

Directions

1. Further integration of informal and formal methods

- **Lightweight formal methods**

- **Involves automated analysis approaches using rapid, low-cost application of FM tailored to immediate project needs [Easterbrook, Lutz, Covington, Kelly, Ampo, & Hamilton; Feather]**

- **Highly selective scope**
- **Limited modeling**
- **Flexible use**
- **Building on existing products**



NASA

RRL-FSE/ICSE00

5

Directions

1. Further integration of informal and formal methods

- **Methodology for appropriate use & application of results to development**

- **Guidelines for support of safety analyses, including of evolving requirements, maintenance**
- **How lightweight?**
- **Reuse within a domain**

- **Integration of previously distinct formal methods [Clarke, Wing, et al.]**

- **Different methods have different strengths, useful for different aspects or phases**
- **Developers can choose level of rigor for analysis of safety-critical components**

RRL-FSE/ICSE00

6

Directions

2. Safe reuse: Safety analysis of product lines

- **Can any safety analysis apply to all derived instances?**
 - **Safety is a property of a specific system [Leveson]**
 - **Exploiting previous analyses requires accurate modeling of dependencies among design options**
 - **Minor variations among systems hard to capture, formalize, verify [Ardis & Weiss, Lutz]**
 - **Incorrect assumptions threaten safety: environment, user, operations**



NASA

RRL-FSE/ICSE00

Directions

2. Safe reuse: Safety analysis of product lines

- **Relationships between software architecture and safety still being defined [e.g., Stavridou; Rushby; Neumann]**
 - **Safety consequences of architectural styles, features**
 - **Architectural solutions that add robustness**
 - **Architectural analysis can support safety analysis of product line instantiation [Gannod & Lutz]**

RRL-FSE/ICSE00

8

Directions

2. Safe reuse: Safety analysis of COTS

- **Does software do what it should?**
 - **Better understanding of constraints on safe reuse**
 - **Assessment of existing product to determine fitness in new “operational envelope” [McDermid] is often difficult and informal**
- **Does software do what it shouldn't?**
 - **COTS may have additional, unexpected behavior**
 - **Formal verification may be used to support certification of interactions**

RRL-FSE/ICSE00

9

Directions

3. Testing and evaluation of SCS

- **Tighter integration of testing tools with safety requirements and analyses [Knight & Nakano]**
 - **Incorporate field data of deployed systems into use cases and test cases [Tsai, Mojdehbakhsh, and Rayadurgam]**
- **Better support of unconventional development processes [Finkelstein]**
 - **Contrary to the textbooks, safety requirements often derived from testing of prototypes**
 - **How to propagate new requirements into safety analysis and verification?**

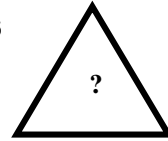
RRL-FSE/ICSE00

10

Directions

3. Testing and evaluation of SCS

- **Evaluation from multiple sources**
 - e.g., Testing, Mathematical review, Certification [Parnas, van Schouwen, Kwan]
 - How to combine evidence is an open problem
- **Testing for consistency between actual behavior and operator's expectation**
 - Model checking for mode confusion [Rushby]
- **Virtual environments**
 - Simulations used for design, testing
 - Tool certification lags usage
 - Human factors complicate understanding of fidelity [Cruz-Neira and Lutz]



RRL-FSE/ICSE00

11

Directions

3. Testing and evaluation of SCS

- **Assumptions about environment**
 - Difficult to correctly identify trigger point at which hazardous state is entered: safe, not trigger-happy
 - Precise environmental modeling essential
- **Assumptions about users**
 - Testing uncovers some mismatches
 - Human factors benchmarks may preclude some mismatches
- **Assumptions about operations**
 - "Test like you fly; fly like you test"
 - Testing requires deep knowledge
 - Operations (and reuse) constrained by tests

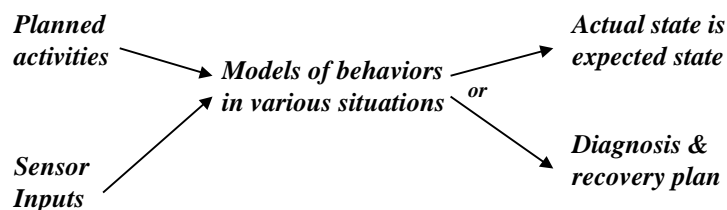
RRL-FSE/ICSE00

12

Directions

4. Runtime Monitoring

- **Detection of hazardous states**
 - **Well suited to monitoring for known, hazardous conditions**
 - **Remote agent software can diagnose broader mismatches between expected and actual behavior and recommend recovery action**



RRL-FSE/ICSE00

13

Directions

4. Runtime Monitoring

- **Profiling of system usage to support safety analyses**
 - **Operational usage inconsistent with safety assumptions**
 - **Evolving conditions that threaten system (survivability)**
 - **Deviations from (safety) requirements and strategies for reconciliation [Feather, Fickas, van Lamsweerde, and Ponsard]**

RRL-FSE/ICSE00

14

Directions

5. Education

- **Few university courses available -> more courses, textbooks**
- **Limited student exposure to SCS -> more case studies, model problems, reading on accidents & their causes**
- **Focus tends to be methodological (how-to) rather than scientific -> build on related work in fault tolerance, security, formal methods, systems engineering**

RRL-FSE/ICSE00

15

Directions

6. Certification and Standards

- **More complicated and less well-defined for SCS systems**
- **100 standards for SCS in 1996 [McDermid]**
- **Issues include [Rodriguez-Dapena]:**
 - **Multinational origin and use**
 - **Composition from many domains**
 - **COTS and legacy components**
 - **Burden of safety case**
 - **Guidance for implementation lacking**



RRL-FSE/ICSE00

NASA

16

Directions

7. Collaboration with Related Fields

- **Security and survivability**
 - e.g., techniques for intrusion detection, noninterference, coordinated responses
- **Software architecture**
 - e.g., safety consequences of product lines, of adaptable architectures, of partitioning
- **Theoretical computer science**
 - e.g., model checking, logics of programs
- **Human factors**
 - e.g., formal specification of mental models, checklists of design features causing mode confusion [Leveson]

RRL-FSE/ICSE00

17

Conclusions

- **Software engineering must continue to exploit advances in related fields to build safer systems**
- **Wider use of safety techniques awaits better integration with industrial development environments**

RRL-FSE/ICSE00

18