RHAS'05 – Paris Workshop Report

12 September 2005

The Fourth International Requirements for High-Assurance Systems Workshop (RHAS'05 – Paris) was held in conjunction with the 13th IEEE International Conference on Requirements Engineering (RE'05) at the Institute of Enterprise Administration (IAE) in Paris France on 30 August 2005.

1 Goals

The workshop addressed the special challenges of engineering the requirements of software-intensive systems, the performance and dependability of which is mission critical. To do this, the workshop brought together in a small focused working group practitioners, consultants, and researchers to:

- Exchange ideas and their experiences concerning the engineering of performance and dependability requirements.
- Identify and explore important challenges and risks.
- Propose, formulate, and evaluate promising solutions.
- Identify open research problems.

2 Workshop Program Committee

The following members of the program committee organized, promoted, and reviewed position papers for the workshop. The workshop chair also facilitated the workshop and ensured the Web publication of the workshop call for papers and proceedings:

1. **Donald Firesmith**, Software Engineering Institute – USA (Chair)
2. Ian Alexander, Scenario Plus – UK
3. Dr. Daniel M. Berry, University of Waterloo – Canada
4. David L. Bush, National Air Traffic Services – UK
5. Dr. Kevin Daimi, University of Detroit – USA
6. Dr. Saeed Fararooy, rcm2 limited – UK
7. Dr. Mats Heimdahl, University of Minnesota – USA
8. Dr. Seok-Won Lee, University of North Carolina – USA
9. Dr. Nancy Mead, SEI – USA
10. Siva Moorthy, Siemens – India
11. Dr. Steve Riddle, University of Newcastle upon Tyne - UK
12. Dr. Guttorm Sindre, Norwegian University of Science and Technology – Norway
13. Dr. Elena Troubitsyna, Abo Akademi University – Finland

3 Workshop Attendees

The following attendees took part in the workshop:

1. David Bush, National Air Traffic Services – UK (speaker)
2. Georgios Despotou, University of York – UK (speaker)
3. Donald Firesmith, Software Engineering Institute – USA (chair, speaker)

4. Tomas Hruby, Eurotel Praha – Czechoslovakia
5. Tim Kelly, University of York – UK (coauthor)
6. Linas Laibinis, Abo Akademi University – Finland (speaker)
7. Seok-Won Lee, University of North Carolina at Charlotte – USA (speaker)
8. Steve Riddle, University of Newcastle upon Tyne – UK
9. Ludovic Robin, IAE de Paris – France
10. Thomas Roosen, IAE de Paris – France
11. Levy Samuel, IAE de Paris – France
12. Alexandra Sebban, IAE de Paris – France
13. Jonathan Weisberg, IAE de Paris – France
14. Nobukazu Yoshioka, National Institute of Informatics – Japan
15. Ruo-Na Zheng, IAE de Paris – France

4 Workshop Agenda

The workshop consisted of the presentation of five position papers and open discussions on topics of interest:

- Introductions
- Morning Brainstorming Sessions:
  - Identification of the characteristics of high-assurance systems that are relevant to engineering their requirements.
  - Identification of the implications of these characteristics on requirements engineering
- Presentation of Position Papers:
  - "Modeling Support for Early Identification of Safety Requirements: A Preliminary Investigation by David Bush, UK National Air Traffic Services Ltd.
  - "The Need for Flexible Requirements in Dependable Systems" by Georgios Despotou and Tim Kelly, University of York
  - "A Taxonomy of Security-Related Requirements" by Donald Firesmith, Software Engineering Institute
  - "Fault Tolerance in Use-Case Modeling" by Linas Laibinis and Elena Troubitsyna, Åbo Akademi University
  - "Engineering Dependability Requirements for Software-intensive Systems Through the Definition of a Common Language" by Seok-Won Lee and Robin Gandhi, University of North Carolina at Charlotte
- Afternoon Brainstorming Sessions:
  - Identification of topics of discussion.
  - Identification of the Impact of Requirements Tracing when Developing High Assurance Systems
  - Identification of the Strengths and Weaknesses of Tradition Requirements Engineering Methods
  - Identification of Sources of Difficulty in Introducing Change in the Way RE is Performed
- Announcement of RHAS'05 – Chicago

5 Important Characteristics of High Assurance System Requirements

During a brainstorming session, the workshop attendees identified the following characteristics of high assurance systems that impact the engineering of their requirements:

- Functionality that is complex and mission critical
- Interfaces that are complex and numerous
- Quality Factors become more important:
  - Accuracy
  - Availability
  - Correctness
  - Feasibility
  - Interoperability
  - Maintainability
  - Performance (including timeliness)
  - Reliability
  - Robustness (including fault-tolerance)
  - Safety
  - Security
  - Sustainability
  - Testability
  - Usability

6 Implications of the Characteristics of High Assurance System on Requirements Engineering

During a brainstorming session, the workshop attendees identified the following implications of the characteristics of high assurance system on requirements engineering:

- Functionality is no longer the most important consideration
- Current lack of sufficient quality requirements of good enough quality
- Need for higher rigor and ability to show that the requirements are correct, consistent, feasible, and verifiable
- Difficulty in obtaining a consensus on the requirements
- Contractual issues make requirements engineering more difficult (e.g., contractual separation of customer, contractor, and subcontractors)
- Configuration management change control of the requirements is more important.
- Interrelationships and engineering tradeoffs between the different types of quality requirements because all quality factors cannot be simultaneously optimized.
- Setting thresholds so that one can determine when one has achieved a sufficient amount of some quality factor.
- Importance in dealing with regulators and achieving accreditation and certification (e.g., safety and security).

- Importance of identifying all requirements and specifying complete requirements (e.g., exceptional paths through use cases, requirements specifying all relevant quality factors and subfactors).

7 Topics for Afternoon Discussions

The workshop attendees identified and voted (3 votes per attendee) on the following topics of discussion:

1. Difficulty in Introducing Requirements Approaches better suited for High Assurance Systems (13 votes)
2. Strengths / Weaknesses of Traditional Methods when used to Engineer Requirements for High Availability Systems (12 votes)
3. Requirements Tracing when Developing High Assurance Systems (11 votes)
4. Size and Complexity of High Assurance Systems (7 votes)
5. COTS/GOTS (4 votes)
6. Non-interference between Requirements due to Complexity, Size, and Multiple Vendors (2 votes)

7.1 Requirements Tracing

During a 20 minute brainstorming session, the workshop attendees identified the following ways that the tracing of requirements is different when developing a high-assurance system:

- **Difficulty tracing quality requirements.**
  Because quality requirements are more important and are not implemented locally, tracing from the quality requirements to their implementation is difficult.
- **Impact analysis.**
  During change control, impact analysis of the impact of requirements on the architecture and design and vice versa may be more difficult, complex, and important. This is especially true with regard to quality requirements.
- **Support understanding.**
  High assurance systems tend to be more complex than other systems, and requirements tracing can help developers and stakeholders better understand this complexity.
- **Tracing to increments.**
  Because of their size and complexity, high assurance systems tend to be developed and released incrementally. Requirements need to be traced to the builds and releases so the developers (especially architects) know what will be needed after the current build or release.
- **Assumptions and rationales.**
  It is more important during high assurance systems to trace requirements to their underlying assumptions and rationales.
- **Need to continue tracing.**
  It is more important during high assurance systems to not stop the requirements tracing task too early.

- **Tracing with COTS.**
  Given the importance of requirements tracing to high assurance systems, how should requirements be traced when dealing with reuse such as COTS, GOTS, OSS (Open Source Software), and legacy software?
- **Tracing with subcontractors.**
  How can you properly trace requirements when dealing with subcontractors who may use different traceability procedures and tools?
- **Specialty engineering.**
  How should you trace requirements when they are specified in specialty engineering documents such as safety and security policies? How can requirements in the different types of repositories and documents be kept consistent? How can traceability help configuration management (change control) apply across these different sources of requirements?
- **Completeness.**
  Requirements tracing can aid in ensuring that the requirements are complete (e.g., ensuring a complete set of derived requirements from higher level requirements and tracing from design to identify missing requirements).

7.2 Strengths and Weaknesses of Tradition Requirements Engineering Methods

The workshop participants identified the following strengths and weaknesses of traditional approaches to RE with regard to engineering the requirements for high assurance systems:

- **Strengths** include:
  - **Characteristics of good requirements.**
    Traditional RE methods clarify the characteristics of good requirements, which are also applicable to requirements important to high availability systems such as quality requirements.
  - **Functional requirements.**
    Traditional RE methods provide reasonable techniques for engineering functional requirements such as use cases.
  - **Requirements management tools.**
    Requirements management tools are even more important to use because of the criticality of managing the requirements of high availability systems.
  - **Traditional elicitation techniques.**
    Traditional elicitation techniques for identifying requirements such as brainstorming, interviews, and JAD sessions are still useful.
- **Weaknesses** include:
  - **Incomplete requirements.**
    Traditional RE approaches typically do not produce a complete set of requirements (e.g., quality requirements).
  - **Requirements ambiguity.**
    Traditional RE approaches often produce architecturally-significant requirements that are ambiguous and insufficiently quantitative, possibly because some

requirements (e.g., quality requirements) are difficult to elicit, specify, and test. This may have more to do with people challenges than technical challenges.

- **Insufficient rigor.**
  Traditional RE approaches often produce requirements that lack of sufficient rigor. Analysis and modeling approaches need to be more rigorous due to the types and criticality of the non-functional requirements.
- **Multiple specialty engineering groups.**
  The important requirements for high assurance systems often require the involvement of multiple specialty engineering groups in addition to RE team (e.g., the safety team, the security team, reliability engineers, performance analysts, etc.). The requirements engineer must work harder in these specialty domains (e.g. to achieve safety and security certifications).
- **Inadequate for development.**
  The output of traditional RE approaches are often inadequate for developers such as architects, testers, users, operators, maintainers, and other stakeholders.
- **Overemphasis on functional requirements.**
  Traditional RE approaches typically emphasize functional requirements and do not give sufficient emphasis to quality requirements and capturing assumptions. Missing assumptions makes it difficult to meet the real requirements.
- **External sources of information.**
  It is difficult for requirements teams to use and maintain the links to external sources of requirements information that were created using techniques not from the requirements engineering community (e.g., the results of hazard analyses such as HAZOP and fault/event trees).

7.3 Difficulty in Introducing Change in the Way RE is Performed

Because of their size, complexity, and importance, developing high assurance systems stresses current development methods to their limits. The working group identified the following reasons why it is difficult to introduce change into the way requirements engineering is performed when developing high assurance systems:

- **Problem not recognized.**
  Although requirements engineering as it is currently practiced has many opportunities for improvement, many developers are not dissatisfied with their current development approaches. They may not recognize the extent of the problem or be in denial as to its magnitude. Many developers are getting away with current state of the practice, either through complacency or lack of knowledge about better practices.
- **Current practices lag best practices.**
  Currently used standards and established processes tend to significantly lag best practices.
- **Lack of maturity and guidance.**
  There is a perceived lack of mature well-documented practices for performing requirements engineering for high assurance systems.

- **Perceived impracticability.**
  The typical requirements engineer does not view more rigorous approaches (e.g. formal methods) as practical.
- **Lack of awareness.**
  The typical requirements engineer lacks awareness of, training in, and knowledge about modern RE approaches that are more appropriate for developing high assurances systems.
- **Lack of green field development.**
  Most current development is enhancement or maintenance of existing systems, which already have requirements that have been engineered using older approaches. It is difficult to justify redoing the existing requirements or to grandfather the existing requirements and live with a mixed set of requirements engineered using two incompatible ways.
- **Admitting imperfection.**
  Some requirements engineers may be afraid to admit that their previously requirements were not optimally engineered.
- **Conservatism.**
  Some managers, technical leads, and requirements engineers are conservative when faced with new technology or processes, which they see as adding unnecessary risk.
- **Hard to convince management.**
  Requirements engineers may find it difficult to convince their management that new processes are either necessary or more appropriate.
- **No prophet recognized in own land.**
  Any requirements engineer attempting to introduce a new process into his or her project may be subject to the inverse of the not invented here (NIH) syndrome. Some managers and engineers believe that only external (and possibly famous) consultants can introduce new processes.
- **Increased short-term cost.**
  Stakeholders may believe that more rigor means more cost, especially if they take a short rather than a long term view. This is especially true if they are unaware of some of the case studies that have shown that the right amount of increased rigor and formality can decrease development costs by greatly decreasing defects and the resources required to fix them.
- **Self interest.**
  Some important stakeholders may believe that changing to new requirements engineering approaches are not in their self interest.