Requirements Models for System Safety and Security

Constance Heitmeyer
Naval Research Laboratory, Washington DC, USA

To capture software system requirements, it is necessary to elicit and analyze the required externally visible behavior of the system to be built within some physical context. Moreover, the system requirements must be modeled and specified in terms of required system services and environmental assumptions. Requirements may be expressed in two general ways – as assertions about the externally visible system behavior or operationally as a model. In either case, the system requirements are represented in terms of environmental quantities. Obtaining high-quality requirements models and specifications is difficult but extremely critical. Poor requirements have been repeatedly identified as a major cause of project cost overruns, delivery delays, and failure to meet the users' expectations of the system.

This course provides a systematic formal approach to the modeling, specification, validation, and verification of requirements for high assurance systems – systems where there is compelling evidence that the system satisfies critical properties, such as security, safety, timing, and fault-tolerance properties. The objective of this approach for a given system is a complete, consistent, and properly organized set of software requirements, including environmental assumptions. The approach is model-based and relies in part on the use of formal specifications for specific kinds of analyses.

The course first introduces basic principles of high quality requirements modeling and specification. Key notions such as "requirements", "model", "environmental assumption", "software specification", "monitored" and " controlled variables", and "system mode" are defined and related to one other. A modeling framework is then introduced in the specific context of requirements for large, complex software systems. Different kinds of models are used to demonstrate and analyze various properties of the system-to-be: among these are requirements models, simulation models, verification models, security models, fault-tolerance models, and testing models. These models are designed for different purposes. For example, the role of a verification model is to support automated checking that an operational model of the system requirements satisfies critical properties of interest. In contrast, the role of a simulation model (or animation) is to help the user validate the requirements specification – i.e., ensure that symbolic execution of the system using a simulator captures the intended behavior.

Models need to be formalized to enable formal reasoning about them. Various specification languages for specifying requirements models are reviewed. Chief among these languages are table-based notations such as Software Cost Reduction (SCR), which allow a user to express the required system behavior as a set of tables. Other common notations are graphical (for example, finite state diagrams) and hybrid (for example, Leveson and Heimdahl's Requirements State Machine Language combines tables and a Statecharts-like notation).

The course will then discuss various techniques supporting the formulation of high-quality requirements, the analysis and validation of requirements to improve their quality, and the use of requirements in obtaining correct system code. Among the techniques covered are

- Modeling and formal specification of requirements.
- Consistency and completeness checking of requirements.
- Simulation of requirements to check their validity.

- Generating invariants from requirements specifications.
- Formal verification of requirements.
- Testing and automatic code generation based on an operational requirements model.
- Modeling and analyzing systems for critical properties (e.g., security and fault-tolerance).

The presentation will be illustrated through representative examples and tool demonstrations.

References

1. C. L. Heitmeyer, R. D. Jeffords, B. G. Labaw. *Automated Consistency Checking of Requirements Specifications.* ACM Trans. on Software Engineering and Methodology; 1996.
2. C. L. Heitmeyer, J. Kirby, B. G. Labaw, M. Archer, R. Bharadwaj. *Using Abstraction and Model Checking to Detect Safety Violations in Requirements Specifications.* IEEE Trans. Software Engineering, Vol. 24(11); Special Issue on Managing Inconsistency in Software Development; pp. 927-948; 1998.
3. R. Jeffords, C. Heitmeyer. *Automatic Generation of State Invariants from Requirements Specifications.* Procs. ACM SIGSOFT International Symp. on "Foundations of Software Engineering"; ACM; 1998.
4. A. Gargantini, C. L. Heitmeyer. *Using Model Checking to Generate Tests from Requirements Specifications.* Procs. 7th European Software Engineering Conf., held jointly with the 7th ACM SIGSOFT Symp. Foundations of Software Engineering (ESEC/FSE'99); LNCS 1687; Springer; 1999.
5. C. L. Heitmeyer, M. Archer, R. Bharadwaj, R. D. Jeffords. *Tools for Constructing Requirements Specifications: The SCR Toolset at the Age of Ten.* Computer Systems Science and Engineering, Vol. 20(1); Special issue on Automated Tools for Requirements Engineering; 2005.
6. C. Heitmeyer, R. Jeffords. *Applying a Formal Requirements Method to three NASA Systems: Lessons Learned.* Procs. 2007 IEEE Aerospace Conf.; 2007.
7. C. L. Heitmeyer, M. Archer, E. I. Leonard, J. McLean. *Applying Formal Methods to a Certifiably Secure Software System.* IEEE Trans. Software Engineering, Vol. 34(1); pp. 82-98; 2008.
8. R. D. Jeffords, C. L. Heitmeyer, M. Archer, E. I. Leonard. *A Formal Method for Developing Provably Correct Fault-Tolerant Systems using Partial Refinement and Composition.* Procs. Formal Methods 2009 (FM'09); LNCS 5850; Springer; 2009.