Software Requirements Specification and System Safety

Mats P.E. Heimdahl

University of Minnesota
Institute of Technology
Dept. of Computer Science
Minneapolis, MN 55455
heimdahl@cs.umn.edu

Jon Damon Reese

Safeware Engineering Corporation
7200 Lower Ridge Road, Unit B
Everett, WA 98203-4925

jdreese@cs.washington.edu

Computer software is playing an increasingly important role in safety-critical embedded computer systems, where incorrect operation of the software could lead to loss of life, substantial material or environmental damage, or large monetary losses. Such diverse technologies as avionics, automobile drive trains, power plants, and medical equipment are relying more and more on the computer to control system parameters. Although software is a powerful and flexible tool for industry, these very advantages have contributed to a corresponding increase in system complexity. Traditional approaches to system development have not successfully handled the problems of increased system complexity. The fatal accidents caused by software in the Therac-25 radiation therapy machine, as well as other incidents, have brought public attention to these problems. Ironically, it is becoming clear that the powerful control logic that software can bring to a system can also impair the ability of the systems analyst to study and understand, and hence safely control, the system's behavior.

Safety is a property of the physical system. It is the physical system that has the ability to uncontrollably release sufficient energy to inflict harm and injury. Thus, safety issues must be addressed during the system design phase. All safety-related functions that are allocated to software must be accurately captured in the software requirements specification and the application experts, for example, systems engineers, must be intimately involved in the requirements specification process; safety issues *can not* be deferred until software development commences. Most software-related accidents have been traced back to errors in the requirements specification rather than coding errors.

In this tutorial we will provide a short introduction to system safety, discuss how the use of software control affects the safety analysis of a system, and outline the root causes of safety-related problems. We will present a formal state-based modeling language called RSML (Requirements State Machine Language) suitable for requirements specification of safety-critical control systems. RSML has been used to capture the requirements for several safety-critical commercial systems, most notably TCAS II (Traffic alert and Collision avoidance System II). Furthermore, using a formal requirements specification language, for example, RSML, enables several types of automated or semi automated analysis techniques that can be used to detect and eliminate potential safety problems from the specification. In this tutorial, we will discuss techniques for automatically detecting incomplete, inconsistent, and nondeterministic requirements, show how fault tree analysis can be used in the RSML framework, and demonstrate how a new analysis technique called deviation analysis can be used to evaluate the effects on the system if the inputs to the system deviates from expected value.

Mats P.E. Heimdahl is an assistant professor of computer science at the University of Minnesota, Twin Cities, and he is Vice President of Safeware Engineering Corporation. Dr. Heimdahl received his Ph.D. in computer science from the University of California, Irvine working with Nancy Leveson.

Jon Damon Reese received his Ph.D. in Information and Computer Science from the University of California, Irvine, under the guidance of software safety pioneer Nancy Leveson. Dr. Reese is the president of Safeware Engineering Corporation and a researcher at the University of Washington.

1090-705X/97 $5.00 © 1997 IEEE