Verifying and Validating Software Requirements and Design Specifications

Barry W. Boehm, TRW

These recommendations provide a good starting point for identifying and resolving software problems early in the life cycle—when they're still relatively easy to handle.

"Don't worry about that specification paperwork. We'd better hurry up and start coding, because we're going to have a whole lot of debugging to do."

How many projects start out this way and end up with either a total failure or a tremendously expensive self-fulfilling prophecy? There are still far too many, but more and more projects are turning the self-fulfilling prophecy around. By investing more up-front effort in verifying and validating their software requirements and design specifications, these projects are reaping the benefits of reduced integration and test costs, higher software reliability and maintainability, and more user-responsive software. To help increase their number, this article presents the following guideline information on verification and validation, or V&V, of software requirements and design specifications:

- definitions of the terms "verification" and "validation," an explanation of their context in the software life cycle, and a description of the basic sequence of V&V functions;
- an explanation, with examples, of the major software requirements and design V&V criteria: completeness, consistency, feasibility, and testability;
- an evaluation of the relative cost and effectiveness of the major software requirements and design V&V techniques with respect to the major criteria; and
- an example V&V checklist for software system reliability and availability.

Based on the above, we recommend combinations of software requirements and design V&V techniques that are most suitable for small, medium, and large software specifications.

Verification and validation in the software life cycle

Definitions. The recent *IEEE Standard Glossary of Software Engineering Terminology*[1] defines "verification" and "validation" as follows:

- *Verification.* The process of determining whether or not the products of a given phase of the software development cycle fulfill the requirements established during the previous phase.
- *Validation.* The process of evaluating software at the end of the software development process to ensure compliance with software requirements.

In this article we extend the definition of "validation" to include a missing activity at the beginning of the software definition process: determining the fitness or worth of a software product for its operational mission.

Informally, we might define these terms via the following questions:

- *Verification.* "Am I building the product right?"
- *Validation.* "Am I building the right product?"

Objectives. The basic objectives in verification and validation of software requirements and design specifications are to identify and resolve software problems and high-risk issues early in the software life cycle. The main reason for doing this is indicated in Figure 1.[2] It shows that, for large projects, savings of up to 100:1 are possible by finding and fixing problems early in the

0740-7459/84/0100/0075$01.00© 1984 IEEE