

4

Software Requirements

4.1 INTRODUCTION

Defined software requirements provide programs and projects with a systematic approach to the development of software requirements provided by various ideas and solutions. Software requirements establish the principals for software design and integration test activities for both software and systems integration. The generation and execution of software requirements are created as a stand-alone item or as an item embedded in higher-level assemblies (i.e., hardware units, workstations, monitor displays, integrated platforms, etc.).

4.2 DEFINITION OF SOFTWARE REQUIREMENTS

Identifying and defining software requirements begin with reviewing the functional or performance requirements developed to identify the constraints on software. The system requirement that is allocated to software evaluations determines accuracy, completeness, and applicability of the requirements for work products.

System requirements allocated to software are refined into greater detail to define derived software requirements. Program and project plans that include the capability to acquire reusable software; software requirements are always identified. The tools (i.e., dynamic object-oriented requirements system [DOORS], matrix worksheets, etc.) can be used for the

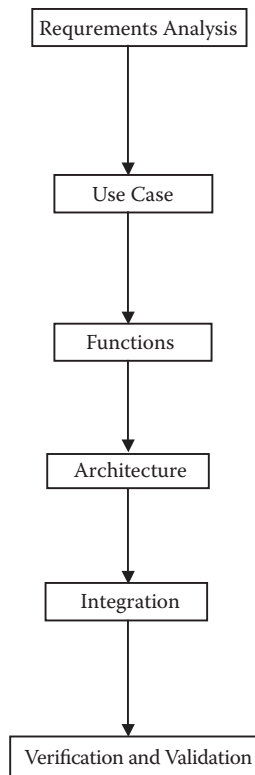


FIGURE 4.1
Software requirements development.

analysis and modeling to gain an understanding of potential architectures and associated software requirements.

The work product for software requirements development is driven by execution and the knowledge of what flows from the start of requirements analysis to verification and validation as shown in Figure 4.1.

4.2.1 Analysis

Requirements analysis includes a step-by-step process to develop requirements for software work products to fulfill high-level user requirements, allocated system requirements, and ideas for system operational concepts. Analysis reports are produced as run procedures are verified and validated to support test and evaluations.

4.2.2 Use Case

A use case is developed to describe a flow of operations for the performance of systems and software implementation. The software use case defines the limitations or technical considerations based on target computers, the execution strategy for a work product, and computer operating systems. Operational cases include functionality, performance, maintenance, and support considerations, as well as the work product's operational environment, including boundaries and constraints.

4.2.3 Functions

The function and architecture definitions are analyzed to ensure an accurate and complete understanding of what software is expected to perform. If program and project plans include reusable software, derived requirements are identified and evaluated for inclusion. Additional knowledge of software operational use cases and the software architecture requires a need to change functionality and associated requirements. This may be an ongoing activity during the software requirements definition process.

4.2.4 Architecture

The architecture interface definition is identified and defined as part of derived software requirements. Graphical representations are prepared and take the form of data flows and software component diagrams.

The functional software design/development life cycle states and modes are established per system requirements. The timing, sequence, conditions, and probability of executing to define and redefine functional interface requirements apply to system architectures.

4.2.5 Integration

The integration of requirements is the transformation of a functional architecture into optimal design solutions. Implementation of disciplined interface management principles is critical for planning resources and to perform systems build integration activities for the execution of meeting software engineering requirements.

4.2.6 Verification and Validation

Software requirements are reviewed to ensure verification and validation. The priority of each requirement is supportive with program and project resources to determine the extent of verification and validation for each defined requirement. The verification and validation for each requirement is identified, and a list of techniques includes analysis, inspections, demonstrations, and tests conducted in software integration facilities.

The most accomplished systems verification and validation of requirements is to plan, evaluate, and record software work product compliance with defined requirements. The risk reduction will assess and ensure software design/development activities satisfy user needs and provide efficient and cost-effective integration of validation and the verification of requirements.

4.3 REQUIREMENTS DOCUMENTATION

The requirements documentation includes software requirements and related information generated, including use cases and derived software requirements, which are the source of each requirement. The software requirements definition is documented according to development plans, software processes, and product standards.

4.3.1 Requirements Traceability

The requirements traceability data are documented according to developing planning, defined processes, and software work product instructions. All software requirements to higher-level requirements or allocated system requirements enter the information into the traceability system according to the program and project's requirements for traceability standards. Software requirements are traceable from system requirements or user requirements and clearly lead to a software architectural component.

4.3.2 Formal Review Preparation

Defined and complete software requirements are critical to have in place before formal review (i.e., functional configuration audit [FCA], physical configuration audit [PCA]) acceptance. Many situations can be discovered

during these formal reviews when requirements are not complete and analysis of these requirements is still open or still in work. The FCA is more involved in reviewing requirements that are more related to the release of software documentation and procedures that trace to hardware drawings and are configured in work products (i.e., systems, software, documentation, facilities, etc.).

From my experience, the systems engineering organization or team is more involved in the FCA, which is required to be completed before PCAs are kicked off and conducted. The customer is involved in both audits; once satisfied and the audits are approved, the customer has the deliverable work product in possession.

4.4 MANAGING A REQUIREMENTS TOOL

Senior program and project managers should look for software requirements tools that meet the following:


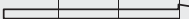




- Ability to impose requirements in multiple formats
- Support for traceability and impact analysis
- Support for software baselines and releases
- Alerts to modifications of the requirements database

Military and aerospace programs and projects utilize many software requirements tools. A most commonly used tool I am familiar with is DOORS. This type of tool can bring order to chaos; I am not promoting this tool. Any effective software requirements tool allows the capability to manage requirements of the same level so software designers can manage source code. I mention and discuss other software tools pertaining to source code and software design/development in Chapter 6.

4.5 RELEASED SOFTWARE REQUIREMENTS

Various released software requirement definition techniques, organizational capabilities, and process guidance capabilities are designed to address the areas of software design/development that can affect

TABLE 4.1
High Failure Rate for Released Software

| | | | | | | | |
|---|---|-----|-----|-----|-----|-----|-----|
| Percent of code that is not executed |  | 20% | | | | | |
| Percent of code that is written but not executed |  | | | 40% | | | |
| Improvement profits when time is for released software |  | 10% | | | | | |
| Potential loss during the software development life cycle |  | | | 30% | | | |
| Total effort expanded due to software rework |  | | | 30% | 40% | | |
| Program failures due to poor requirements |  | | | | | 60% | |
| Percentage for high failure rate for released software | 0% | 10% | 20% | 30% | 40% | 50% | 60% |

requirements definitions. Directly addressing these areas to align business goals and objectives reduces rework, increases productivity, and ensures that requirements lead directly to program and project success and effective software deliveries to customers. Numerous factors contribute to the high failure rate for released software, but the most significant factor is related to poor and undefined requirement gathering, analysis, and management. The high failure rate for released software is defined in Table 4.1.

FURTHER READING

Ambler, S., 1995. *Software Development, Using Use Cases*. Cambridge University, Cambridge, UK.

AS9100, 1997. *Aero Space (AS) Standard Quality Management System Requirements—Guidelines for the Application—Part 2*.

Gonzales, R., *Requirements Engineering*, 2004. Sandia National Laboratories, <http://www.incose.org>.

Kazman, R., and A. Eden, January 2003. Defining the terms architecture, design, and implementation, *News at SEI*, 6(1). Software Engineering Institute, Pittsburgh, PA.