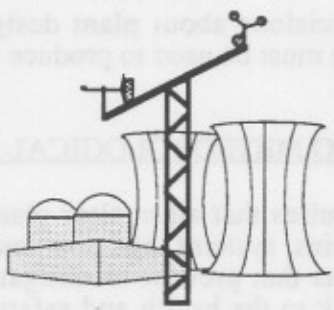
SOFTWARE QUALITY ASSURANCE AT TVA

by
Kenneth G. Wastrack

Tennessee Valley Authority
Atmospheric Sciences
Muscle Shoals, Alabama

NUMUG



NUCLEAR UTILITY METEOROLOGICAL DATA USERS GROUP

Charlotte, North Carolina

October 1994

Software Quality Assurance at TVA

The nuclear power industry is highly dependent on computers as integral parts of its functions. Not only are computers used for day-to-day activities, but computer-generated data is often critical in many engineering design decisions.

Well-established engineering standards exist for computers and related hardware. However, the software used on these computer systems is often not subject to an equivalent level of scrutiny. Software problems that have caused breakdowns in long-distance telephone service and air traffic control systems are well known. More recently, problems with the software that controls the automated baggage handling system are delaying the opening of Denver's new international airport [1].

Software problems may not appear to be a major issue in providing meteorological support. However, the Tennessee Valley Authority (TVA) has found that reliable software is necessary to produce several meteorology-related products for:

- Joint frequency distributions of wind direction, wind speed, and atmospheric stability for routine dose calculations.
- Chi/Q calculations to make estimates of potential dose impacts on the public resulting from accidents.
- Climatological summaries for design applications, and as information for regulatory applications (FSAR, etc.).
- Cooling tower drift and deposition estimates.

Important (and often expensive) decisions about plant design are based on data from these applications, so high-quality software must be used to produce these products.

REQUIREMENTS APPLICABLE TO METEOROLOGICAL MONITORING

10 CFR Part 50 Appendix B [2] requires that all nuclear plants have a quality assurance (QA) program for the testing of the structures, systems, and components of the facility. This includes "structures, systems, and components that prevent or mitigate the consequences of postulated accidents that could cause undue risk to the health and safety of the public." As part of plant systems, software applications have a prominent role in the nuclear industry and are critical in many aspects of nuclear plant licensing, design, and operation.

To address this area, a TVA Nuclear standard [3] identifies software QA requirements applicable to TVA's nuclear power plants. Within the scope of this standard, a software quality assurance plan [4] defines and describes the specific QA requirements that apply to meteorological support activities. This software quality assurance plan (SQAP) is the basis for the rest of this paper.

TVA APPROACH

The TVA approach to software quality assurance as it applies to meteorological support activities is an evolving process. It is not now, and probably never will be, a strictly defined "cookbook" process. The information presented in this paper is best described as a status report that presents current TVA ideas, an examination of how they have worked in practice, and some lessons for the future.

The basic principle guiding the SQAP is independence of the three basic functions; owner (prepares the specifications), developer (prepares the design and writes the computer code), and reviewer (performs the verification and validation of the computer code). This independence helps ensure that a single viewpoint does not drive the software development process and helps provide an objective appraisal of the adequacy of the final product. The individual functions are performed independently until they must merge in the final stage to achieve a consensus about the completed product. Several people may perform these basic functions and specific individuals may participate in more than one function, but different people direct each basic function (owner, developer, and reviewer).

Like any other QA requirement, documentation is a vital component of the process and, except the executable software code itself, is the most important result from the process. The SQAP requires several documents as evidence the software application performs as intended (table 1).

THE SOFTWARE DEVELOPMENT PROCESS - STEP-BY-STEP

Figure 1 illustrates the SQAP process that conforms to the Software Development and Testing Cycle illustrated in figure 2. Note that the references in this description to owner, developer, and reviewer refer to the function and not specific individuals.

Step 1: Prepare Requirements Specification

The owner prepares the Requirements Specification to describe exactly what the software application shall do. This involves clearly defining all applicable assumptions and requirements. The failure to define an assumption or requirement frees the developer to handle it in any way desired.

Documentation = Requirements Specification.

Step 2: Prepare Verification/Validation Plan

The reviewer prepares the Verification and Validation (V&V) Plan based on several software testing principles (table 2). This step begins as soon as the desired results are identified, since feedback from the preparation of the V&V Plan can influence how the requirements are specified.

Documentation = V&V Plan.

Step 3: Prepare Design Specification

The developer prepares the Design Specification. The Design Specification provides the technical details on implementing the Requirements Specification. Historically, the Design Specification takes the form of flow charts. In satisfying the SQAP, the Design Specification still consists of flow charts, but also includes more detailed information concerning specific aspects of the program.

This step does not begin until after preparation of the V&V Plan has started. This permits tailoring the design of different parts of the software application to address likely acceptance test configurations.

Documentation = Design Specifications.

Step 4: Verification - Design Evaluation

The reviewer verifies the Design Specification satisfies the Requirements Specification. This verification step identifies design errors before translation into computer code (while they are still easy to correct). The primary function is to ensure that the design addresses all requirements. However, it is also important to ensure that the design does not include features not described in the requirements. This reduces the risk of using the software application inappropriately to take advantage of an undocumented (and consequently untested) feature.

Documentation = Included in test results or a separate document as specified by the V&V Plan.

Step 5: Perform Software Coding

The developer performs software coding of the application based on the Design Specification. Historically, this is the heart of the development process for computer software and has always been considered more art than science. In the SQAP, this step is an engineering activity because the Design Specification describes all the program aspects in detail. In addition, as more software is written according to the SQAP requirements, a library of fully tested modules will be developed. Approved modules will then be available for new applications so new code will not need to be developed.

Documentation = Software Code Listing (extensive comments are assumed).

Step 6: Verification - Software Code Evaluation

The reviewer verifies that the Software Coding satisfies the Design Specification. This is usually a line-by-line code walkthrough to ensure that the design is properly implemented.

Documentation = Included in test results or a separate document as specified by the V&V Plan.

Step 7: Validation - Software Code Testing

The reviewer validates the software application works as intended using a series of tests. This is the heart of the SQAP process and is performed according to the V&V Plan. Three different levels of tests--unit, integration, and system--document that all components work as intended (figure 3). If problems are found, the software code is reworked and retested until test results are acceptable.

Documentation = Test Results.

Step 8: Develop User Documentation

The developer prepares User Documentation. User Documentation is not necessary to document that the software application works as intended. However, if any user other than the developer executes the program, clear consistent instructions are necessary. Even the developer may occasionally need the prompting that the User Documentation provides.

Documentation = User Documentation.

Step 9: Installation, Check-out, and Final Acceptance

The owner, developer, and reviewer functions merge in this step to concur that the completed software application performs as required. The software application is not implemented until individuals responsible for all three functions agree and signoff that the software application is acceptable.

Documentation = Final Acceptance.

The steps outlined above are not discrete activities performed and completed before the next activity can begin. Often, later steps in the process identify a change that is necessary in some earlier documentation. For example, a test result may identify the need for a design change or an option in the User Documentation may require changing the Requirements Specification. The SQAP provides for flexibility in performing different steps by not requiring documentation to be in final form until final acceptance. This permits changes in all documents during the development process as necessary. However, after Final Acceptance, changes are not allowed to any documentation except according to a revision process that parallels the development process for the original software.

The SQAP does not specify the format of the documentation, only the purpose and the required information. Documentation can be combined into as many or as few individual documents as necessary.

THE COST OF SOFTWARE QUALITY ASSURANCE

Software quality assurance is not inexpensive. It is estimated that the cost to develop a software application with 1000 executable lines is about $50,000 (which translates to $50 per line or $2.50 per character) as broken down below:

Complete Requirements Specification	$ 5,000
Complete Verification/Validation Plan	$ 5,000
Complete Design Specification	$ 4,000
Perform Verification of Design Specification	$ 2,000
Perform Software Coding	$ 5,000
Perform Verification of Software Coding	$ 8,000
Perform/Review Validation Tests	$14,000
Complete User Documentation	$ 2,000
Final Acceptance	$ 5,000

The traditional costs of software development (Design Specification, Software Coding, and some Validation Tests) represent only about 25 percent (~$ 12,500 in the example above) of the total, so roughly 75 percent of the cost represents the costs of QA documentation. This leads to an important question:

Is it justified to increase software development costs by 4 times to prepare documentation?

In purely economic terms, the answer is negative. But there are many potential hidden costs that may occur if adequate documentation is not available. These include:

- Learning Costs--This includes costs to determine how the software application works when the original developer is not available. Often this type of situation occurs with a short turn-around and has a high potential to produce errors that can cascade into other problems.
- Reworking Costs--This includes costs for reworking software because the developer did not properly understand technical details of the activity addressed by the software application. This often happens because different individuals inherently have different interpretations based on their experience and technical background.
- Client Costs--Data output is rarely the ultimate product of a software application but is used (either directly or indirectly) to make engineering analyses. Errors in the output data can increase costs significantly if new engineering analyses indicate the need for significant (i.e., expensive) changes to plant systems or structures.

The additional expenditures beyond the traditional costs for software development are justified as an investment to lessen costs in the future. While the costs to perform the QA functions are not inconsequential, they are still much less than the costs to repeat work to fix problems.

A CASE STUDY - XOQ145

Nuclear Regulatory Commission Regulatory Guide (RG) 1.145, "Atmospheric Dispersion Models for Potential Accident Consequence Assessments at Nuclear Power Plants," identifies acceptable methods for assessing the potential offsite radiological consequences for postulated releases of radioactive material to the atmosphere. TVA developed a computer program (XOQ145) to satisfy the requirements of RG 1.145. This is a summary of that project.

Autumn 1992

TVA decides to modify its existing Chi/Q program to handle elevated releases along with ground-level releases. Since the involved staff are inexperienced in software QA documentation and this is the first application to implement the SQAP requirements, it will serve as a prototype for future work.

The completed computer code will contain about 1000 executable lines, with an estimated base cost of $50,000. Since the program is a modification to an existing application and will make extensive use of existing code, the estimated total project cost is reduced by 25 percent to $37,500. The project should be completed in 4-5 months.

Only about $17,500 is immediately available for the project, so it is decided to only prepare the Requirements Specification, the Design Specification, and perform as much Software Coding as funds will allow. Testing is deferred until more funds become available.

The decision to defer testing represents the first major mistake:

The development and testing organization is not established at the start of the project.

The owner, developer, and reviewer should be identified at the start of the project so that all participants can agree on exactly on the bases and the goals of the project, and how the results will be tested (at least in general terms). The Requirements Specification, the Design Specification, and Software Coding activities can then be conducted to facilitate testing and verification. Since the complete development and testing organization was not established at the start of the project, the owner and developer performed their work in a piecemeal manner that did not fit together into a single package. This emphasizes the second major mistake:

The format for the quality assurance documentation was not predetermined.

XOQ145 was a prototype for future work, so no prior examples of software documentation were available. Consequently, the owner and the developer prepared documentation according to their own preferences. When the testing process did start, each tester prepared individual documentation. A significant effort was necessary at the end of the project to rework all the information into a consistent package.

Summer 1993

The initial versions of the Requirements Specifications, the Design Specifications, and the Software Coding are complete. Funds are now available for testing. A reviewer is selected and completion of the project is expected by the middle of autumn 1993.

At this point, testing is viewed as a relatively simple process involving a quick review of the documentation and executing a few test cases to verify the Software Code.

This was the third major mistake:

Testing of software applications was expected to be a short and simple process.

The magnitude of testing was greatly underestimated because of a lack of experience in performing quality assurance activities. Fortunately, the reviewer was extremely knowledgeable about software development and was quickly able to correct misconceptions. Among the first revelations was that the task of adequately testing the software application was beyond the capability of one individual to perform within a reasonable time. Therefore, additional staff was assigned to testing the software application. Because the new staff had not been previously involved in software testing in general or this specific project in particular, a significant amount of time was necessary to orient them on the XOQ145 project and to provide training on testing techniques. In addition, it was now recognized that previously existing code needed to be tested since it had not previously been covered under the SQAP. Thus, XOQ145 was a new program in every way.

Winter 1993-94

The Requirements Specification and Design Specification have been revised to address issues raised by the verification steps. These documents, the V&V Plan, and the Software Coding are essentially complete. Only Testing and development of User Documentation remain. The initial deadline of autumn 1993 has passed, but since there is no urgent need for the XOQ145 program, the additional time can be accommodated.

There is now adequate staff to develop test cases in an expeditious manner, but a significant further delay is occurring because only one person is executing the test cases on the computer. This is creating a minor backlog in getting results back to the test case preparers for review. The software application should be operational in Spring 1994.

Two problems in the testing process stretched the time of the project. The first problem was that the staff preparing test cases was not assigned on a full-time basis, but did the work as other work permitted. This meant that participant focus was lost in submitting test cases and reviewing results. The second problem was that the effort to execute test cases was underestimated. This meant that test cases were not executed expeditiously and returned to test case preparers for review. Consequently, the test case reviewers needed extra time to re-aquaint themselves with the test cases before performing the reviews. The time delays were relatively small for individual test cases, but since approximately 180 different test cases were required, the time delays accumulated.

Spring 1994

Testing is completed and the software application is declared to be operational. There are some inconsistencies in documentation formats, so revisions are still necessary to improve the clarity of the documentation. However, this does not restrict the use of the XOQ145 program. Formal signoff and implementation will take place during the summer of 1994 after the documentation is revised.

Another underestimate was made in the time it took to correct documentation deficiencies. All test cases had to be re-identified and rerun so that the program outputs contained proper identification headers. A few test cases were revised to correct minor errors, but no program errors were found. In addition, minor documentation modifications were made to distinguish the original material from future revisions. While essential, the effort to prepare clear and consistent documentation was still time consuming. Had a documentation format been established early in the project, the documentation could have been prepared in a consistent format from the start and would not need to be revised later.

Fall 1994

The revision of the documentation formats is complete. However, changes in the program assumptions are now necessary. In addition, preliminary runs of the completed program have calculated some values that do not compare with similar data from other sources. It will be necessary to fix the program assumptions and evaluate program output against other data sources. Finally, a maintenance plan needs to be developed to address future program changes.

The XOQ145 application will be completed by the end of 1994.

After Requirements Specification were complete, the SQAP did not involve the owner in the main activities of the project until the project was ready for Final Acceptance. Consequently, the need for format changes was not identified until the program was ready to be declared operational. Back-tracking and redoing work caused more time delays.

The XOQ145 project, originally estimated to cost $37,500 and take 4-5 months to complete, has cost $60,000 (table 3) and taken over 2 years to finish.

The XOQ145 project was an excellent test case. The SQAP will be revised to address several issues that were identified. Among the specific issues are:

- The requirement for independence of the key functions will be relaxed to allow personnel to be better utilized. For example, had the XOQ145 owner been more involved during the middle of the project, formats might have been defined at an earlier stage and thus reduced the need for extensive efforts at the end of the project.

- The SQAP will specify assembling of the project team at the start of the project and definition of project goals before any significant work can begun.

- The SQAP will include checklists for each function to provide direction in performing work.

KEY LESSONS FOR SOFTWARE QUALITY ASSURANCE

TVA's experience with software quality assurance has taught the following lessons:

- Unless software has been previously tested in a well-documented manner, treat every project as a new application requiring full documentation. A history of using a software application for many years without problems does not guarantee that it will work as intended in another application.
- Set up the development and testing organization at the start of the project. Not only does this identify who will perform each function, but it ensures that all participants know the bases and goals of the project.
- Establish documentation formats at start of the project. Participants will know what the final product should look like and consequently will reduce the effort to standardize formats at the end of the project.
- DO NOT underestimate costs. The biggest mistake that can be made is not providing adequate resources to complete the project. Some aspects of the project may not be performed properly or delays may occur.
- Set up a plan for maintenance of the completed software application. Even if no changes are anticipated in the application, a maintenance plan is necessary to ensure it is always available and works as designed. For example, if the software application is moved to another computer system, it should be retested to ensure that differences in operating systems will not cause problems.
- All test cases should involve two people -- one to prepare and one to review. There is no 'cookbook' methodology for software testing, so a second person is valuable to help ensure that important tests are not missed.
- Perform all software QA activities within a relatively limited timeframe. Whenever possible, software QA activities should be concentrated in time to keep the staff focused and reduce the need for re-orienting staff if they have to share time with unrelated activities.
- Never assume commercial software works as expected. Too often, it is assumed that commercial software will always work as part of a new software application. Commercial software, that will be part of a software application, should be tested to ensure that it performs as expected within the particular software application. TVA has experienced two cases when commercial software did not work correctly. In one, a FORTRAN compiler provided by the computer hardware vendor didn't work. In the second, a spreadsheet program did not perform some calculations correctly.

CONCLUSION

Software quality assurance is not a luxury. In many cases, software quality assurance is essential to perform many tasks and make critical decisions. While the cost of software quality assurance is high, the cost of not having software quality assurance can be prohibitive.

Corollary to Murphy's Law

It is impossible to make anything foolproof because fools are so ingenious.

ACKNOWLEDGMENTS

I would like to thank a number of individuals without whose effort the XOQ145 project would never have been finished. Hollis Lindley wrote the preliminary version of the XOQ145 program. Benjie Norris educated everyone on the meaning of software testing. Larry Gautney and Mary Jacobs prepared and reviewed numerous test cases. Sherea Burns executed the test cases and provided valuable assistance in correcting problems. Their efforts and sacrifices are greatly appreciated.

REFERENCES

1. "Software's Chronic Crisis," Scientific American, September 1994

2. 10 CFR Part 50 Appendix A, "Quality Assurance Criteria for Nuclear Power Plants and Fuel Reprocessing Plants," 1975

3. TVA Nuclear Power standard STD-2.12, "Control of Computer Application Software," 1993

4. "Software Quality Assurance Plan for Meteorological Support to Nuclear Power," AS-QAP-3
   - revision 0, October 1, 1991
   - revision 1 (draft)

Table 1 - Documentation Required by TVA Software Quality Assurance Plan
(from TVA's "Software Quality Assurance Plan for Meteorological Support to Nuclear Power")

Document	Prepared by	Purpose
Requirements Specification	Owner	Delineates all requirements that the software application must satisfy.
Design Specifications	Developer	Technical description of how the software will satisfy the Requirements Specification.
Verification & Validation (V&V) Plan	Reviewer	Describes the evaluation methodology to document that the software application properly carries out the requirements.
Software Code Listing	Developer	Computer language listing of the software application that makes heavy use of comments to describe the code functions. Headers are used to describe subroutine calls and variable definitions.
Test Results	Reviewer	Documents that the completed software satisfies the evaluation criteria of the V&V Plan.
User Documentation	Developer	Describes function and operation of software application for users.
Final Acceptance	Owner Developer Reviewer	Documents concurrence that software application is complete and performs required function.

Table 2 - Software Testing Principles
(from The Art of Software Testing by Glenford J. Meyers)

A necessary part of a test case is a definition of the expected output or result.

A programmer should avoid attempting to test his or her own program.

A programming organization should not test its own programs.

Thoroughly inspect the results of each test.

Test cases must be written for invalid and unexpected, as well as valid and expected, input conditions.

Examining a program to see if it does not do what it is supposed to do is only half the battle. The other half is seeing whether the program does what it is not supposed to do.

Avoid throw-away test cases unless the program is truly a throw-away program.

Do not plan a testing program under the tacit assumption that no errors will be found.

The probability of the existence of more errors in a section of a program is proportional to the number of errors already found in that section,

Testing is an extremely creative and challenging task.

Testing is the process of executing a program with the intent of finding errors.

A good test case is one that has a high probability of detecting an as-yet undiscovered error.

A successful test case is one that detects an as-yet undiscovered error.

Table 3 - Cost Summary for XOQ145 Project

Activity	Estimated Cost	Actual Cost	Difference
Complete Requirements Specification	$3,000	$5,000	$2,000
Complete Design Specification	$4,500	$4,000	($500)
Software Coding	$10,000	$6,000	($4,000)
Complete Verification/Validation Plan	$2,000	$6,000	$4,000
Perform Verification	$3,000	$15,000	$12,000
Perform/Review Validation Tests	$7,000	$16,000	$9,000
Complete User Documentation	$3,000	$2,000	($1,000)
Final Acceptance	$5,000	$6,000	$1,000
TOTAL	**$37,500**	**$60,000**	**$22,500**

Figure 1
Development of Application Software
(as described in the SQAP)

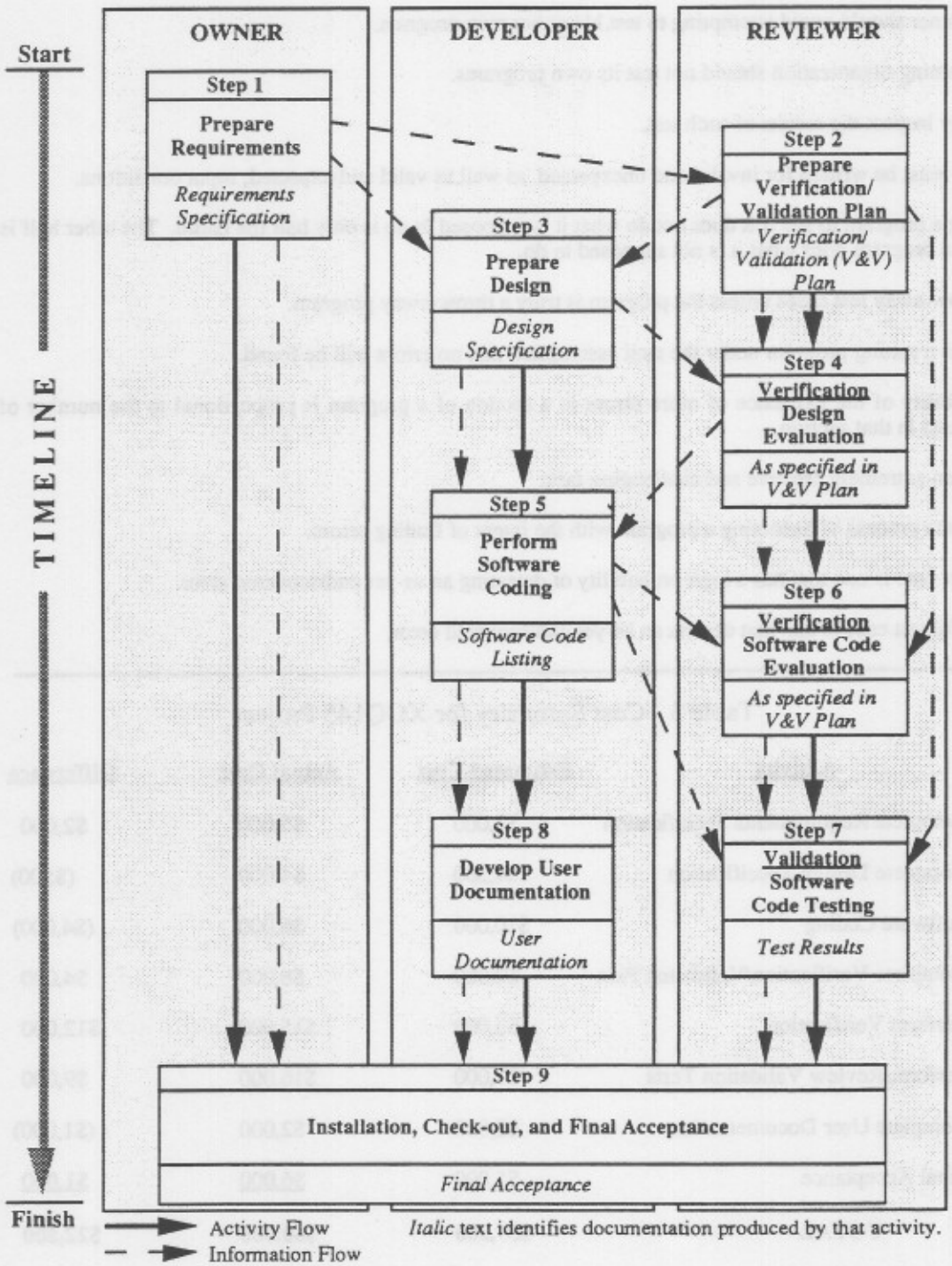


Figure 2
Software Development and Testing

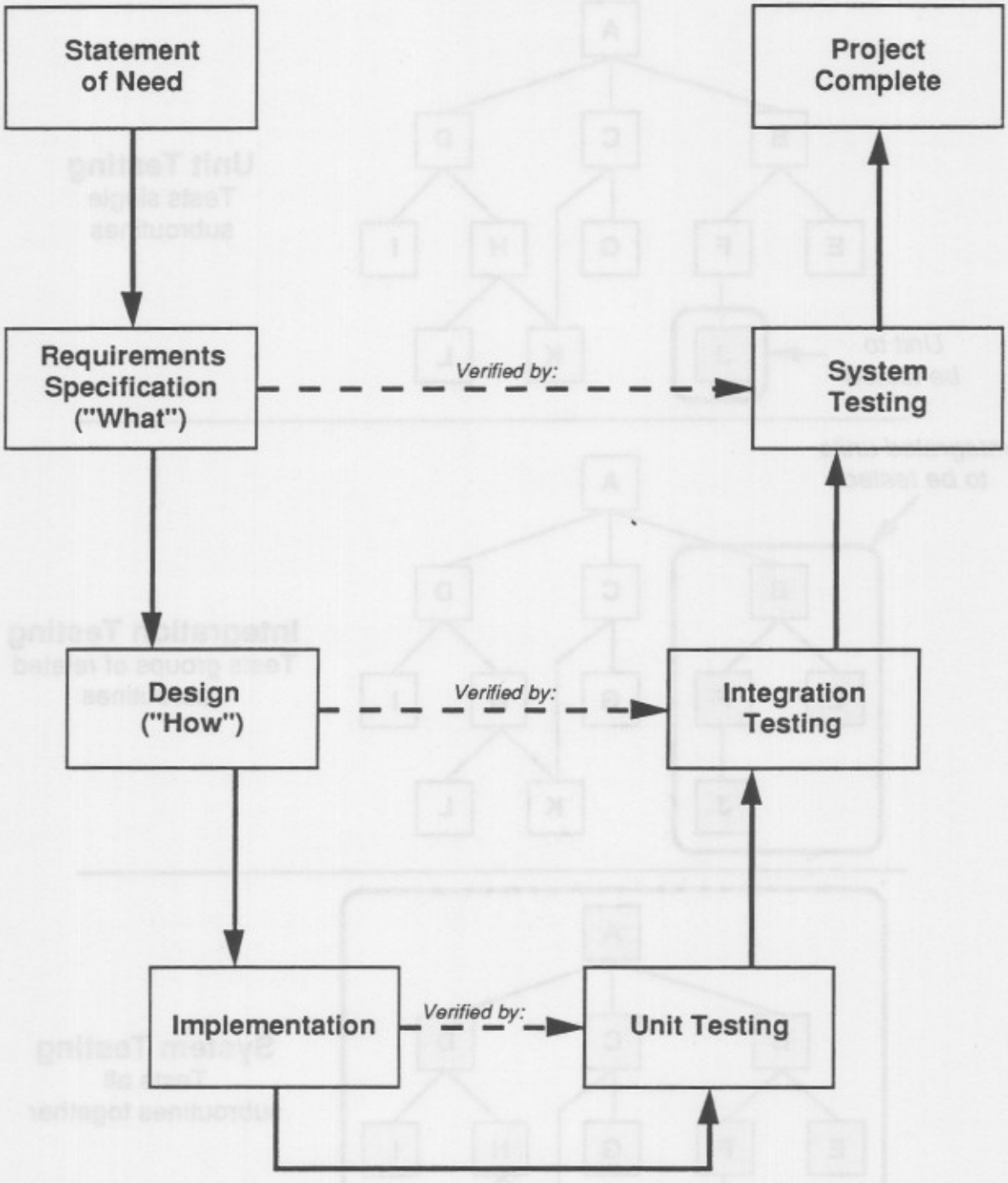


Figure 3
Levels of Testing