Customer Satisfaction Through Quality Software

Dan Coats, Hewlett-Packard
Michael McCaffrey, Hewlett-Packard

Perhaps the most frustrating thing for a Hewlett Packard customer is updating to a new release of MPE. The majority of the older customers take the "wait and see attitude" before updating their production systems to the new software because they have been burned in the past. A few brave souls and new customer installations take on the task of being the guinea pigs. At a point, maybe 3 to 6 months after release, when the software has stabilized, the rest of the customer base is feeling more comfortable and gradually migrates onto the new software. This attitude, though justifiable, creates a number of problems for Computer Systems Division (CSY). Since the next release of MPE is based on the most current one in the field, we need a high percentage of customers using the current release to insure that we are not carrying problems forward into the next release. When bugs are found and fixes are generated for the installed customers, they are then incorporated into the next version of the product. Because of the time lag between the release of new software and the installation of it on a high percentage of the installed systems, we are working with a unknown quality level of software. What can be done about this CATCH 22 situation? We believe the most important thing is to raise your confidence level in our software. The only way to do this is to provide the highest quality software possible with the current technology available. With the stage set at this point, this paper will discuss some of the things we are doing at CSY to address this goal. This paper will first define the Product Life Cycle, which is a mechanism for defining the process of software development. The quality checks and testing at each phase will be covered in addition to what we are doing to automate the system testing to uncover as many problems as possible before software is released to our customers.

Software Product Life Cycle

The software development process is controlled by a document called the Software Product Life Cycle. This document defines the objectives, results or output, and method of validation for each phase of the life cycle.

This discussion will deal with a generic life cycle. While not exactly like the life cycle used at CSY, it defines the mechanism that most software divisions model their Product Life Cycles after, including CSY. There are six major steps in the process of developing software.

- o Investigation The requirements for the product to be successful in the market are determined. The product objectives, in terms of functionality, usability, reliability, performance, and supportability are defined.
- o Design External interfaces, both user interfaces and product interfaces, are defined. Internal specifications, which state the algorithms that will be used, are developed.
- o Implementation Code, module test, and document the product components. Plans are made for field and user training.
- o Testing Insure that the product meets HP and user requirements. Integration of the software. System testing, ALPHA and BETA testing. Field review and supportability evaluation.
- o Release Package and transfer to manufacturing and the field.
- o Support Ongoing enhancement and product fixes.

Software Product Life Cycle Testing Activity

In the early steps of the Software Product Life Cycle a great deal of testing activity occurs. The following chart shows the testing activity broken down for each step of the Product Life Cycle.

Phase	Product Development	Testing Activity
Investigation	Product Requirements	----------
Design	External design, and Internal designs	Design Inspections Quality Plans Test Plans
Implementation	Coding	Code Inspections Module Testing
Testing	Integration of the software	Function Testing System Testing ALPHA & BETA Sites Field Review

Quality Plans

The quality plan is a document that states the quality objectives of the project and the means for achieving them. These measurable objectives and specific plans will come from all functional areas and will result in goals agreed upon by the entire product team. As the project progresses, the Quality plan is used for evaluation and planning throughout the life of the product.

The quality plan is not a test plan, nor does the test plan deal with the functions of a quality plan. The scope of the quality plan is much broader, its creation and acceptance must coincide with those of the External Specifications in order to insure that the various aspects of the product's quality are considered at appropriate points in the product's life. A test plan, however, defines specific testing activity, is much more detailed and is created later in the product's development.

In order to successfully define the quality objectives of a product, the plan addresses each phase of the product life cycle and includes the following:

-specific objectives to be met;

-the means of achieving these objectives;

-what measures will be used to determine the degree of quality achieved;

-criteria needed before project enters the next phase;

-what documentation will be kept for all the activities.

The goals should be expressed in terms of fitness for use. There are five major aspects of fitness for use and each should be addressed for each phase of the project. These are:

-functionality

-usability

-reliability

-performance

-supportability

Quality Plans

Contents

The following table identifies the major components of the Quality Plan:

I. Product Overview

A. Product Identification

B. Related Products and Projects

C. Quality Perspective of Product

II. Plans by Phase

A. Design

B. Implementation

C. User Acceptance Testing

D. Release to Manufacturing

E. Post-Release

Quality Plans
I. Product Overview

A. Product Identification

1. Name, mnemonic, and number

2. Product Abstract

3. Project Personnel

B. Related Products and Projects

A list of all products and projects that affect this product is provided in order that dependencies among products can be identified.

C. Quality Perspective of Product

A description of the product emphasis, its market, and its potential end use is provided. In addition, problems that crop up as a result of the related products and project investigations are recorded and contingency plans are developed as well as methods for monitoring the problem areas.

II. Plans by Phase

For each of the five development phases objectives, plans, and methods of verification should be described. The quality control procedures should reflect specific measures to be taken to insure a high quality result at the completion of each project milestone; the quality improvement section focuses on what items or processes will be improved as a means of improving the resultant product's overall quality; the measurement section focuses on how the project team will know if its goals have been met. This outline will look at the objectives only; plans and verification methods will be dealt with later.

A. Design

The design phase consists of external and internal subphases which may overlap, depending on the needs of the project. In the external subphase the features of the product and the user interface are specified. The internal subphase defines the structure that will support the product's externals. During this phase, a test plan is also developed by the project team. This plan will be described later.

1. Objectives

External....

--to describe the product operating environment

--to determine the functional capabilities

--to define the user interface

--to specify documentation standards for the ES

Internal....

--to describe major elements of the internal structure

--to develop complete, detailed descriptions of the algorithms and data structures to be used in the im plementationplementation

--to provide design alternatives

B. Implementation

This phase includes the coding of all modules, the completion of the internal maintenance specification, testing of the product, and manual writing.

User reference manuals, programming guides, and other manuals are completed so that they will be available for both alpha and beta test sites. This is an area known for a large number of product errors, therefore, particular care should be taken to review the manuals to ensure they are accurate, complete, and useful.

1. Objectives

Coding.....

--to determine coding standards to be used

--to provide idea of internal documentation expected

--to decide if hooks should be included for testing

--to faithfully translate the algorithms and data structures developed in internal design into programs, procedures, and data structure definitions

--to provide complete and functionally tested code

Internal Maintenance Specification.....

--to produce concurrently with the code

--to provide a complete description of the internal design, structure, and flow of a product

--to detail the development of the product

--to provide the information necessary to support and enhance the product

--to provide sufficient information so that field training may be developed

--to provide for evolution of the document after the product is released

Testing Activities.....

--to develop tests as outlined in the test plan

--to document tests in test documentation section

--to choose ALPHA test sites

C. User Testing

This phase exposes the product to a controlled end user environment. In order to get to this phase, all lab testing is completed, and the product has completed the implementation phase.

1. Objective

--to evaluate the initial reliability of the product

--to determine readiness for BETA site testing

--to provide active test site management

--to test the product installation files

D. Release

During this phase, the code must be tested extensively to insure its reliability, the performance must be measured and tuned until the product is ready for customer use, and control of the product must be turned over to manufacturing.

1. Objective

--to provide for thorough product testing through the use of the BETA test site

--to stress the product with the intent of causing it to fail

E. Post Release

The life of a product after release involves changes, both fixes and enhancements, the necessity to interface with new products, and the requirement to minimize disruption of existing customers' activities.

1. Objective

--to demonstrate that new releases have improved quality as measured by agreed upon metrics

--to test and prove that the code of the new release is maintainable

--to insure that documentation of the new release meets quality objectives

--to insure that advantages of fixes or features in the new release outweigh the risk of potentially introducing new bugs

--to reduce the time that a critical or serious problem remains open

The test plan is a strategy for applied testing of a product. The objective of a test plan is to define the scope, resources, and timetable of product testing. All of these will vary depending on the type of product and the objectives documented in the quality plan. The quality

plan makes specific recommendations about the testing process- the test plan implements them.

The test plan is the driving force for the testing process and as such, should address the following:

--the types of tests to be used by the project team

--non-standard test specifics

--frequency of repetitive operations (e.g., walkthroughs, regression tests).

It should define a set of tests which will be sufficient to guarantee the quality of the finished product upon release.

The test plan is developed as a integral part of the design and implementation phases; it is directly applied to the external specification.

The following is an outline of a test plan that is presently the standard for CSY. The intent, when developing such a plan, is to use only those areas that are directly applicable to the product being tested and this paper will only deal with some of the aspects of such a plan.

Test Plans Contents

I. Product Identification

A. Project name, mnemonic, and project number

B. Project Abstract

C. Project Personnel

II. Development Quality Control Practices

A. Design Walkthroughs

B. Design Reviews

C. Code Inspections and Walkthroughs

D. Code Reviews

III. Implementation Testing

A. Module or Unit testing

B. Integration testing

C. Development test tools

IV. Product Testing

A. Function testing

B. System testing
1. Facility
2. Volume
3. Stress
4. Usability
5. Security
6. Performance
7. Storage requirements
8. Configuration
9. Compatibility/Conversion
10. Installability
11. Reliability
12. Recovery
13. Serviceability
14. Documentation
15. User procedures
16. Certification

C. Acceptance testing

D. Installation testing

E. Test automation

F. Equipment and configuration needed for testing

I. Product Identification

This section is self-explanatory.

II. Development Quality Control Practices

In order to promote quality in the design and implementaion phases, the test plan delineates the ground rules and frequencies of walkthroughs, inspections, and design and code reviews.

III. Implementation Testing

A. Module or Unit Testing

Attempts to find the discrepancies between the external description of the module and its logic as demonstrated in executing the code.

B. Integration Testing

This involves combining the next module to be tested with the set of previously tested modules before it is tested. This tests the interface between two modules and the logic of one of them when the other has been previously module tested. The order used in integrating modules can simplify the testing process and have other important consequences.

C. Development Test Tools

IV. Product Testing

A. Function Testing

Looks for the discrepancies between the external functions as described in the external specifications and what the product actually provides.

B. System Testing

The plan selects which of the types of tests should be run; defines the objective and what verification criteria will be applied for each of the tests.

C. Acceptance Testing

This is the process of comparing the product to its initial requirements and the current needs of its end users. ALPHA and BETA test sites are the prime example.

D. Installation Testing

When installing many software products, a variety of options must be selected by the user; files and libraries must be allocated and loaded, a valid hardware configuration must be present, and the product must be interconnected to other products. Installation testing is the way to weed out these anomalies. This testing also occurs at the ALPHA and BETA sites.

E. Test Automation

F. Equipment and configuration needed for testing

Inspections, both design and code, are formalized processes used to find defects. The main idea behind inspections is to reduce or eliminate bugs as early as possible in the development cycle, thus increasing programmer productivity and lowering the product life cycle costs. Additionally, inspections provide a technically correct base for the next phase of the development cycle and insure adherence to the product specifications. The inspection team consists of four to five people who are assigned the following roles:

Moderator This is the key person for a successful inspection. The moderator is responsible for managing the inspection team, scheduling a suitable meeting place, reporting the inspection results, and following up on any rework.

Designer The person responsible for producing the design or code.

Reader Any team member other than the moderator or designer. Responsible for reading the material, one line at a time. Also has the responsiblities of an inspector.

Inspector Responsible for finding defects in the material being inspected.

The material to be inspected is distributed at least one week before the inspection is scheduled. Along with the material is the responsibility assignment for each member of the team. The formal inspection begins with the reader reading the document aloud, one line at a time. As defects are found, the moderator classifies and records them. Examples of such classifications are design logic errors, coding errors, interface errors, etc. The moderator faces two problems when the defects are found; one is to keep the remarks depersonalized and the other is to avoid solving the problem during the inspection. Keeping the remarks depersonalized is not an easy task. It is difficult for people not to say "you didn't..." or "you should have..." when they are discussing something they think the designer should have done differently. Instead, they should say "the design is wrong..." or "the code is...". The other thing to keep in mind is that the inspection is held to find defects. The resolution of the defects should be handled outside of the formal inspection meeting. The moderator should follow up on all defects to ensure they are corrected. The entire inspection should last no longer than two hours. It has been found that after two hours, the error detection efficiency of most inspection teams begins to dwindle.

A module, used in the structured programming sense, is a procedure or closely related set of procedures. If you look at two different approaches to software integration, you begin to see why module testing can be very important.

"Big Bang" Approach

1. Design, code, and test each module by itself.

2. Throw all the modules into a large bag.

3. Shake the bag very hard.

4. Cross your fingers and hope that it all works.

Big Bang Approach makes it very difficult to track down a bug. Which module is causing the error?

Incremental Approach

1. Design, code, and test one module by itself.

2. Add another module.

3. Test and debug the combination.

4. Repeat steps 2 and 3.

Incremental approach allows the process of debugging to be more scientific. Using the

incremental approach requires the use of stubs, drivers or both, when testing of a module begins. The first module is tested using dummy external calls. When the next module is ready, the dummy external call is replaced by a call to the new module. The tests against each module are saved and rerun with the addition of each new module. The incremental approach has allowed us to find things like interface errors and variable initialization problems very quickly. This not only simplifies the debugging process but produces a higher quality product in the end. Automated System Testing On any new release of MPE, there are over 800 job streams, containing over 10,000 separate tests, run against the operating system and the associated subsystems. In the past, operators worked around the clock, streaming each job and checking the output results for errors. Although this method did get all the tests run, one was never sure that all the errors were found and reported. Therefore, the decision was made to develop a series of tools to automate the testing process, in the hope of catching more errors earlier in the system testing process. The first program to be developed was the monitor process. This program is script driven and controls the streaming of all test jobs. We use user logging to record any errors in the testing process, and have written a set of intrinsics to allow the tests to report the P-relative code location of an error. In addition, system traps are armed by the monitor process to catch any abnormal aborts of the system intrinsics. With the controlling process written, the next step was to write a program to save the output spool results for later analysis. This program, called the ARCHIVER, is created by the MONITOR and copies all spool files to a tape using SPOOK. With the MONITOR and ARCHIVER in place, we began to catch a greater number of the test errors earlier in the testing process. The next problem was who was going to read thru 80,000 lines of spool output looking for inconsistencies. To solve this, a spoolfile comparision program was written. We took a known good run of the tests and checksummed them into a MPE file. The ARCHIVER now creates the spoolfile comparision program and passes each completed spoolfile to it for analysis. The spoolfile is checksummed and compared against the MPE file created from the known good run of the tests and errors such as missing lines, additional lines, and lines that do not match are reported. The automated testing has allowed us to catch a large number of defects in the software that would otherwise have reached the customer. Projects are now in process to enhance the automated testing for more functionality and to increase the testing coverage of the operating system.

Bibliography

Software Product Lifecycle. Hewlett-Packard; SPLC.83.02 #5955-1756; February, 1983.

Software Quality. DAS Quality Management Team, Information Networks Division, Hewlett-Packard; July, 1981.

A Guide to Writing Quality Management Plans. SNA Quality Management Team, Information Networks Division, Hewlett-Packard; August, 1981.

Quality Plan. Memo from Sally Dudley, Hewlett-Packard; May, 1980.

An Overview of Hewlett-Packard's Software Life Cycle Preliminary from Sally Dudley, Hewlett-Packard; September, 1983.

Design Inspection Seminar Jim Dobbins, IBM; January, 1983.