Software Engineering Institute

An Acquisition Perspective on Product Evaluation

Grady H. Campbell, Jr.
Harry Levinson
Richard Librizzi

October 2011

TECHNICAL NOTE
CMU/SEI-2011-TN-007

Acquisition Support Program
http://www.sei.cmu.edu

Carnegie Mellon

* These restrictions do not apply to U.S. government entities.

Table of Contents

List of Figures

Acknowledgments

The authors would like to thank the Department of Defense (DoD) programs that allowed us to observe the many positive and negative examples that enabled us to create this technical note. We would also like to thank the following for their technical and editorial assistance: John Foreman, John McGregor, and Murray J. Connelly.

Executive Summary

The Software Engineering Institute's (SEI) experience with a variety of DoD acquisition programs has shown the need for improvements in acquisition and sustainment product evaluation practices. This technical note proposes a comprehensive approach to product evaluation that will reduce acquisition and sustainment costs by increasing the quality of results and eliminating the need for repeated testing and rework.

Current practice emphasizes testing as the principal means of determining whether a product meets specified customer needs and quality criteria, but testing is time-consuming, expensive, and inconclusive. Testing inherently occurs late in the development of a product when discovery of defects requires rework; rework undermines the architectural integrity of the product resulting in further testing, additional defects, and further rework.

Further experience in industry has shown that a variety of proven practices can augment and reduce this narrow dependence on testing; these include precise acceptance criteria, early validation of product requirements to those criteria, and a disciplined iterative development process for continuous product integration. Others include more rigorous, technically focused reviews, performed continuously throughout development. This will better ensure a systematic effort to reduce rework through earlier detection of defects and an emphasis on a product line approach for eliminating redundant efforts. Instituting these practices would provide the basis for more efficiently acquiring products that will be of sound quality, responsive to the customer's needs, and cost-effective in the face of increasingly constrained budgets.

Abstract

This technical note focuses on software acquisition and development practices related to the evaluation of products before, during, and after implementation. From engagements with numerous DoD acquisition programs, it has been observed that a number of recurring issues reduce the effectiveness of how software-reliant products are evaluated. An acquisition effort consists of identifying the customer's needs, selecting or developing a product that is responsive to those needs, and then evaluating the product to determine if it properly addresses the identified needs. This technical note describes the Product Evaluation (verification, validation, and certification) process including test, reviews, and formal methods. It also makes the argument that Product Evaluation should not be deferred until after a product has been built, but should begin as soon as the customer's needs have been identified and should continue throughout the acquisition effort.

1 Introduction

Acquisition is a systematic effort to obtain products whose capabilities will improve the ability of an enterprise to perform its mission. An acquisition effort consists of identifying the customer's needs, selecting an existing product or selecting a supplier to develop a new product that is responsive to those needs, and then evaluating that the as-built product properly addresses the identified needs. An acceptable product is deployed into use as part of an operational system and over time is sustained to continue satisfying the needs for which it was built.

The effort of evaluating a product for acceptability should begin as soon as the customer's needs have been identified, but not deferred until after a product has been built, and should continue throughout the acquisition. Evaluation requires gaining and maintaining a proper understanding of the customer's needs, advocating development practices that will reduce the frequency of defects, and planning actions to identify and diagnose any flaws in the product while it is being built. Early detection of defects is less costly both in terms of the effort required to make corrections and in terms of maintaining product integrity. *Product Evaluation* (verification, validation, and certification) encompasses all of the means that can be used to determine whether a product is going to meet a customer's needs—including testing, reviews, and formal methods.

The SEI has observed a number of recurring issues—in performing engagements with numerous DoD acquisition programs and their associated development contractors—concerning how software-reliant products are evaluated. This note focuses on issues related to how acquisitions approach the evaluation of products during and after development and suggests various potential improvements in these practices.

1.1 Observations on Common Practice

More than half of a typical acquisition effort is spent evaluating the product that is being obtained. The purpose of these evaluations is to determine whether the product is being built properly and whether it provides the capabilities that a customer needs [Boehm 2001, Bennett 2005].

Much of the evaluation effort during product development is indirect, involving reviews of extensive, as-built documentation without needed characterization of alternatives and tradeoffs considered or rationale for subsequent decisions. Without an understanding of how these considerations have influenced product development, evaluation reduces to a determination of consistency and completeness relative to perceived customer needs rather than whether the product is a best fit to those needs given development constraints of cost, schedule, and technology.

After product development, evaluation reduces to anecdotal testing: effort focuses on finding instances for which product behavior may differ from expectations in representative cases of operational usage. Differences, when determined to be defects in the product, trigger an indeterminate but rushed period of rework during which the product is incrementally revised to work properly given the chosen cases, but often also introducing new defects to be found and corrected.

At the root of this costly and ineffective approach to product evaluation is an initial description of the customer's needs that is typically poorly organized, inconsistent, incomplete, and yet prematurely over-specific in constraining potential solutions. Uncertainties about actual needs and expectations for potential future changes are seldom communicated. Acceptance criteria cannot be precisely articulated with such a poor foundation, but is expressed instead in the form of test cases that reduce broadly stated needs to necessarily narrow cases that are only representative examples of those needs.

Reviews and testing will remain the primary means that acquirers and developers use to evaluate the acceptability of a product; however, we can hope to improve these by looking more closely at what constitutes proper evaluation of a product. In doing this, we need to consider several questions:

- Can product evaluation efforts be applied during product development in a way that will significantly reduce discovery during product acceptance of defects that cause delay and the need for costly rework?
- What is the optimum level of product testing? How much testing is needed to ensure that a product is acceptable for use?
- Is testing effective for finding all of the defects that could reasonably be found? Are defects being found that should have been found earlier, in reviews or prior testing?
- Are there techniques other than testing and reviews that can be used in order to increase confidence in a product earlier or with less effort?
- How can the effectiveness of work product reviews be improved to ensure product quality and reduce defects that would be found later through testing?
- When a product previously found to be acceptable is modified, does the product need to be comprehensively reevaluated or can the effort needed to determine that the modified product is also acceptable be reduced?
- In situations where multiple versions of a product are needed—tailored for different uses, operational contexts, or technologies—can evaluations be leveraged in order to minimize per-version evaluation costs?

1.2 The Nature of Product Evaluation

There are various views of a product, each expressed through a different representation [Kopetz 1976]. These representations must be mutually consistent, with each comprising a partial model of the product that answers only certain questions about it:

- *Customer needs*: expressing the customer's expectations concerning what capabilities the product should provide
- *Operational context*: describing the physical and information environment in which the product operates
- *Acceptance criteria*: defining the customer's criteria (threshold and objective) for determining whether the product is acceptable to be put into use
- *Requirements*: specifying the product's expected behavior

- *Design*: specifying the architecture, constituent components, and prescribed organic and emergent properties of a product that is a satisficing (satisfy and suffice) solution for the requirements
- *Implementation*: constituting a concrete realization of each of the product's components
- *User documentation*: defining how the product is operated and used as part of an organization's business/mission enterprise

The relationships and differences among these representations are illustrated in the following example:

- Customer needs specify "no loss of critical data"
- Operational context specifies information that is accessible to the product from its environment
- Acceptance criteria specify what data is critical, that any data over 30 seconds old must be recoverable within 30 seconds, and that any resulting data gaps must be logged and reported to the system manager
- Requirements specify a product information model (including critical data items) and specify retention and recovery policies that satisfy critical data retention acceptance criteria
- Design specifies a data integrity process and components for dynamic and persistent storage; it specifies policies and interfaces for committing and recovering or rolling back data to a consistent state and for detecting and handling data discrepancies.
- Implementation provides the components that realize the designed approach for ensuring data integrity
- User documentation describes, from a user's perspective, what to expect in terms of data currency and what happens if a gap occurs

The role of product evaluation in acquisition is to validate and elaborate the product acceptance criteria in the form of reviews or tests that are then used to determine whether each of the other representations properly and consistently satisfy that criteria. Through the process of evaluating each of the product's representations, the product is progressively shown as satisfying the customer's needs. When discrepancies arise, the cause must be traced to its source in one or more of the product's representations and corrected to reestablish mutual consistency.

1.3 Limits to Testing as a Product Evaluation Method

With the emphasis given to testing and the level of effort expended on it during an acquisition, it is natural to assume that this is the best, or only, means that an acquirer can rely on to be sure of getting an acceptable product. Testing alone, however, is not an adequate technique for evaluating a software-reliant product; the limitations of testing are widely recognized and must be effectively augmented with other methods of evaluation [Linger 1996; Spillner 2007].

It is not possible to determine conclusively whether a product has only prescribed behavior and is free of defects. Testing can only show—in the best case—that attempted trials have produced expected results and have revealed no defects. The infinite combinatorial variety of potential valid and invalid inputs and internal data/processing states of a product makes exhaustive testing impossible; only a representative coverage of possible inputs and states is feasible within a finite schedule and budget. The most that testing can achieve is a level of confidence that testing performed is representative of expected actual usage and that defects will not arise in operational uses that do match the cases tested. This confidence can be undermined if different users follow different usage patterns that cannot all be tested.

Not only is exhaustive testing not feasible, there is not yet a commonly accepted objective measure for determining when testing efforts have been sufficient to stop and declare the product acceptable for use. Again, the best option is to determine what criteria will be used to judge whether testing has provided an adequate level of confidence that all critical defects have been discovered. The best means for doing this is to design test sets that provide, based on estimated defect density, a statistically sufficient level of code and path coverage, input data boundary condition coverage, and scenario-based usage coverage.

A particular impediment to testing is the necessity that an operable product exists to be tested. This means that testing finds defects at the time when rework is most costly. Whereas many types of defects may be discovered using other techniques—such as reviews or analytic models that are applicable to incomplete products, systemic defects, or emergent properties such as performance or reliability—defects can be discovered with testing only through use of the product under specific conditions. These defects can be a result of fundamental architectural decisions that require significant effort to change.

In addition, many types of processing logic are difficult to test, requiring initial computational conditions that are difficult to replicate. In particular, concurrent forms of processing (including multi-processor and multi-threaded, timing- and event-dependent, and distributed latency-sensitive logic) are not understood well enough to predictably reproduce and diagnose complex defects.

Testing must ensure not only that the product behaves correctly under normal conditions but also that it behaves acceptably in the face of unexpected operational conditions (including failures). Again, the most feasible approach is to categorize potential unexpected conditions and failures as the basis for test sets that cover representative cases.

Software operates within the physical constraints of hardware; results of testing on different hardware can produce different results depending on the software's sensitivity to hardware characteristics. Slow processors can limit the software's functional capabilities; fast processors can expose undetected conflicts in concurrent resource/data sharing. Usage of needed hardware resources (e.g., displays, data stores, sensors, effectors, communications) can be restricted if other software also needs access to those resources at the same time. Different hardware can represent data with different degrees of accuracy, precision, or value limits. Results of tests can change if operational hardware differs from test hardware or if any changes are made in the operational hardware. Similarly, product behavior, as well as corresponding test results, can differ if the behaviors of interacting systems are inaccurately modeled in testing or vary in unforeseen ways. Analogously, training modes in the product's behavior can exhibit these same issues to the degree that inaccurate models are substituted for inaccessible parts of the actual environment.

The remainder of this note presents a more complete view of product evaluation as the practice within an acquisition effort for determining whether a product is going to satisfy the stated needs of a customer. Section 2 gives an overview of the elements of product evaluation, whereas Sections 3-5 discuss (in more detail) the preparation for and performance of evaluations during and after product development. Section 6 discusses product evaluation in the context of a product line acquisition and Section 7 suggests specific actions that can be taken to improve product evaluation practices in general.

2 Understanding Product Evaluation in an Acquisition Context

Viewed broadly, acquisition encompasses product development and product acceptance (Figure 1). *Product development* entails obtaining a usable product that targets identified customer needs, whereas *product acceptance* entails determining whether the obtained product is a proper fit to those needs. Product evaluation is an essential element of both of these.

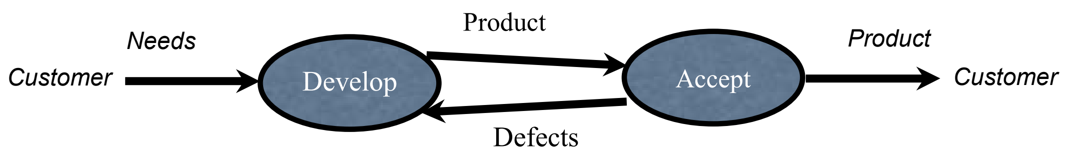

Figure 1: The Product Acquisition Cycle

More specifically:

- Product Development consists of the iteratively performed activities of systems and software engineering (requirements, design, implementation, and integration) required in creating a product. The role of product evaluation during development is to verify that the product is being correctly developed (i.e., consistently across all work products as specified) and that it is converging on specified customer acceptance criteria.
- Product Acceptance consists of integrating the product with other system elements and performing product evaluation to determine whether the product behaves correctly in its expected usage. The product must be verified against customer acceptance criteria and validated that it provides the customer with the operational capabilities actually needed; any divergences from expectations must be diagnosed and characterized as defects for disposition in a subsequent iteration of product development.

Product evaluation begins—at or before the start of product development—with the formulation of customer acceptance criteria. This representation is an explicit and objective characterization of acquirer-specified customer needs and gives product development and product acceptance efforts a shared view of those needs. Although, in the end, a product will be judged acceptable only if it behaves operationally in the way that the customer expects, the definition of formalized acceptance criteria helps to ensure that expected behavior has been communicated accurately. Equivalently, this provides an early basis for systematically identifying and correcting any omissions, ambiguities, or misunderstandings concerning the customer's needs as communicated. With this basis, product evaluation becomes an objective means for determining whether a product should be accepted for deployment and use.

During product development, product evaluation must be performed both collaboratively with engineers and evaluators and independently to ensure a quality product that will meet customer acceptance criteria. During product acceptance, product evaluation should be chartered as exclusively representing the interests of the acquisition and customer organizations. Further, product evaluation should be coordinated across product development and product acceptance to ensure that development and acceptance efforts are working toward satisfying commonly understood customer acceptance criteria as the authoritative definition of targeted customer needs. Development and acceptance activities can operate concurrently, iterating through product versions until the product has been determined to be acceptable for customer use.

2.1 Product Evaluation Methods

Although testing is an essential element of product evaluation, it is not the only (or always best) means of establishing the acceptability of a product [Linger 1996]. Effective product evaluation relies on a mix of methods that can ensure early, least-effort discovery of defects for correction. These methods broadly include reviews, testing, and formal methods, augmented by runtime methods that can limit the impact of undetected defects.

2.1.1 Reviews

The most cost-effective technique for product evaluation is reviews by a mix of mentors, peers, and subject matter experts [Pomeroy-Huff 2009]. While testing requires an operable product as its subject, reviews can occur anytime during development as various work products are produced. By properly reviewing each work product, defects can be discovered and corrected before work has even started on other work products that might depend upon it. Product evaluators looking across related work products can discover inconsistencies in how developers understand the customer's needs and the technical approach being taken and how these inconsistencies can be resolved to avoid defects. To be most effective, reviews should focus each reviewer on specific technical questions, particularly emphasizing areas that the author views as difficult or uncertain.

2.1.2 Testing

Testing is an important means of product evaluation because it builds directly upon knowledge of how an enterprise operates [Perry 2000; Bhor 2001; McGregor 2001; Spillner 2007]. However, testing is also necessarily an anecdotal method in that it is impossible even to enumerate, much less test, all of the conceivable ways that a product might be used under all conceivable conditions. To mitigate the impossibility of exhaustively testing a product in all the ways it may be used, a strategy of layered progressive testing is widely used. Testing proceeds in layers corresponding to the logical structure of the solution, the assumption being that testing at each level will be effective at discovering defects that occurred at the corresponding stage of development.

- Component: The product design specifies a set of components whose implementations comprise the base units from which a product is constructed. Each component is independently tested to confirm that its implementation conforms to its design.
- Product: The product being acquired is constructed by selecting and integrating verified components in accordance with the design. The product as a whole is then tested, following scenarios that approximate how the product will be used, to verify that its behavior conforms to its requirements.
- System: The product is integrated with other elements of the customer's operational environment to verify that the product behaves as expected, as specified in the customer acceptance criteria.

- Operational: The integrated system is validated as behaving according to customer expectations in light of current operational needs.

Testing is a means to evaluate a product based on specific scenarios and test cases that represent each tester's understanding of how the product will be used. Testers must select scenarios and cases that seem representative of the inputs and actions that the product is most likely to encounter in actual use, as well as cases that represent less common or abnormal uses but that could expose defects with more significant detrimental effects.

Even with a sound means to identify cases that might have been missed, testing alone is not going to discover every defect in a product. The challenge, then, is determining how to focus testing efforts to get the best results and how to use other methods to augment product quality. Getting better results means having higher confidence that fewer significant defects remain to be discovered and corrected later. Significant defects are those that have the greatest impact on correct behavior of the product, particularly those that limit effective performance of the customer organization's mission.

2.1.3 Formal Methods

In addition to reviews and testing, options are emerging for greater use of formal methods that can reduce the amount of time spent on evaluations or to expose defects that are otherwise difficult to discover. Formal methods include techniques such as static analysis, static or symbolic testing, assurance cases and model-based dynamic analyses for verifying quality factors, and domain-specific, correct-by-construction engineering methods [Weinstock 2004; Feiler 2010].

Reviews and formal methods are particularly relevant in the context of product lines where testing of individual products alone would provide inadequate leverage. Reviews and formal methods that can be applied to an abstractly represented set of similar, yet-to-be-developed products provide the means to verify satisfaction of criteria and expose defects across an entire set of similar products.

2.1.4 Runtime Strategy

Although a systematic effort to use a variety of methods to detect defects can improve the quality of products, no combination of methods can guarantee the complete absence of defects. To provide a final level of protection, there should be a runtime strategy and mechanisms implemented in the product to detect and handle failures due to undiscovered defects, limiting their potential impact and retaining information that will enable subsequent diagnosis and correction. A part of product evaluation is to ensure that this strategy and its associated mechanisms are in place.

2.2 Product Evaluation Activities during Product Development

Product evaluation during development has the purpose of helping to reduce product acceptance efforts and delays that result from late discovery of defects requiring rework. By helping developers discover (understanding and construction) errors as early as possible, the effort required to correct those errors and to maintain consistency among all work products is minimized.

Product evaluation activities during product development consist of

- Reviewing customer acceptance criteria to ensure that customers' needs, as currently understood, are being properly communicated
- Reviewing requirements to ensure consistency with customer acceptance criteria and other available information concerning business process, operational environment, and future customer needs
- Reviewing design (architecture) for consistency and completeness relative to requirements and proper consideration of design alternatives and tradeoffs
- Reviewing formal design-and implementation-level models that provide analyses of evidence for satisfaction of specified quality factors
- Reviewing design (component specifications) for consistency and completeness relative to architecture and conventions
- Reviewing internal designs of components for consistency and completeness relative to their specifications
- Reviewing component implementations for consistency and completeness relative to their specifications and internal designs
- Advising and assisting in the construction of component and integration test materials (using relevant elements of the product testing infrastructure as appropriate) and reviewing component and integration test results for evidence of defects
- Developing and performing integration tests to ensure that components work together to provide an operable product, identifying and diagnosing any discrepancies
- Developing and performing product-level behavioral testing to verify conformance with requirements (to identify and diagnose any behavior that diverges from expectations)
- Analyzing the causes of any identified defect to determine whether improvements in development or evaluation practices could avoid or detect such defects earlier

For increased effectiveness, product evaluators should be collaboratively paired with product developers to help identify and clarify any misconceptions concerning customer needs and to improve the quality of developer verification efforts. Product evaluators that assist multiple developers can help ensure consistency of understanding and solution approach among them. The goal of product evaluation during product development is to discover and eliminate defects through the active involvement of increased domain knowledge and continuous verification. Although a development environment may not permit exact replication of operational conditions, awareness of potential differences can help to isolate the scope of the impact of those differences on the product.

2.3 Product Evaluation Activities during Product Acceptance

The purpose of product acceptance is to ensure that a product will behave as needed and expected in the specified context of its operational use. Product acceptance has two aspects:

- Verification: determining that the product is a correct solution to the problem as understood
- Validation: determining that the product properly addresses the customer's needs

Product evaluation for product acceptance occurs at two levels: system and operational. System evaluation begins with system integration—installing the product with other hardware and systems to create a facsimile of the complete operational environment. The product is then verified as operating properly in that environment—following representative business process scenarios—in accordance with customer acceptance criteria. Operational evaluation validates whether the product will satisfy specified customer needs when used by representative users in performing specified business processes in the actual (or a closely representative) business operational environment.

2.4 Orchestration of Product Evaluation Activities

Performing product evaluation requires more than the direct efforts of reviewing, analyzing, and testing the product and constituent work products; it requires setting up a framework that is conducive to performing product evaluations effectively and efficiently. This requires additional effort, entailing several activities that can be performed prior to or concurrent with primary product development and product acceptance activities:

- Augmenting requirements derived from customer needs, by specifying required capabilities and instrumentation that will enable the observability that is needed for product evaluation activities
- Planning activities, staffing, and resources that are needed to perform product acceptance
- Building a test environment by replicating or emulating the product's specified operational environment
- Acquiring operational data and/or tools that can be used to generate appropriate test data
- Acquiring and tailoring the tools needed for analysis and reporting of observed versus expected product behavior indicated by test results
- Acquiring and installing required operational hardware and/or emulators to support testing
- Development of operational scenarios, which consist of tests that emulate normal and exceptional product use, and definition for each test of expected results, which characterize the product's expected behavior
- Building and installing the product—with appropriate instrumentation—to operate in the test environment
- Performing tests to collect results and operational profile data
- Analyzing test results and operational data in order to diagnose, report, and track defects indicated by differences in expected versus observed product behavior

3 Prerequisites to Product Evaluation

The role of product evaluation in acquisition is to determine whether a product that is being acquired is suitable for operational deployment and use. The objectives of this effort are to ensure that the purpose of the product is properly understood, that corresponding criteria for its acceptance by the customer are clear, and that the product meets those criteria.

Effective product evaluation begins from a proper definition of the customer enterprise and the activities of that enterprise that the product is meant to support. A proper definition of an enterprise includes knowledge of the business objectives, roles, processes, and procedures that the enterprise performs. The concern for product evaluation with this is that the enterprise definition properly informs the intended purpose of the product.

3.1 Acceptance Criteria vs. Requirements

Acceptance criteria and requirements are alternative representations of customer needs that the product is meant to satisfy. They differ in that the acceptance criteria define the minimal capabilities that the product must exhibit to be acceptable to the customer, whereas the requirements—after refinement through analyses of engineering alternatives and tradeoffs—characterize the observable behavior of a specific solution. From a product evaluation perspective, requirements must specify product behavior that will satisfy customer acceptance criteria and then the product as-built must satisfy these stricter requirements. During product acceptance, product evaluation must only verify that the product, which has been shown during development to meet validated requirements, as a whole does in fact conform to the acceptance criteria.

For product evaluation to be effective, product development methods must anticipate the techniques that will be used to verify intermediate work products. In preparation for product development, product evaluation should ensure that the methods to be used, and any supporting automation, provide for observability of work product content and include documentation of assumptions, alternatives considered, and rationale for choices made.

A particular challenge in choosing development methods is achieving a balance between describing the capabilities that are needed in a product and prescribing a specific solution. Some prevailing techniques tend to promote concrete, overly specific descriptions because these are more easily conceived and understood from the perspective of a user. Examples of this are often seen in the common practice of defining requirements as free-form “shall” statements or as more stylized use cases. These and other such techniques benefit from a user’s ability to describe how they work or how they expect a product to behave. The drawback of these more prescriptive forms of specification is that they depend upon users making assumptions and choices that prematurely eliminate potentially better alternatives. Furthermore, the actual variety of ways and conditions in which a product may be used precludes describing in detail all of those conceivable uses and the result is that the set of such descriptions is unavoidably incomplete; while more abstract forms of description exist, it is more difficult for people to understand and envision implications of such descriptions until a final product has been developed.

The goal of product evaluation is to encourage practices, regardless of method, that distinguish adequately between essential characteristics of a needed product and those that are only included for sufficient descriptive completeness to enable shared understanding. The best choice may be a mix of a more formal notation that supports precise specifications and an associated, less formal notation that is more anecdotal and overly detailed as an aid to understanding, with the formal notation being authoritative.

3.2 The Testing Environment, a Development Effort in its Own Right

To provide valid results, testing requires an environment that is an adequate approximation of the actual operational environment. This requires an operable form of all the elements with which the product interacts, including both the equipment that comprises the computing infrastructure and other systems (adversarial systems, natural forces, unintentional operator actions, etc.) and devices that operate in that environment and communicate or share resources with the product. If operational versions of other systems and devices are not available, their behavior must be simulated.

Just as the customer's needs are the basis for building the product, the basis for building the testing environment is the customer's description of the product's operational context. Operational context defines the nature of the operational environment into which the product will be deployed. If the customer's characterization is not properly understood and adequately realized in the test environment, evaluations of the product may lead to incorrect conclusions regarding the quality and acceptability of the product.

In addition, both the environment and the product must be configurable and controllable as needed to perform the various test scenarios. This includes a mechanism to initialize the data state of the environment and the product to represent a prescribed initial state for each test scenario. The testing environment must also support instrumentation that enables the profiling of internal behavior and the collecting of data about the product's behavior that is needed to perform quantitative and qualitative analyses of the product's properties.

The testing environment is a framework for evaluating not only the operable product, but also the documentation and support that are needed by users of the product. This documentation must be further augmented with information on the use of the testing environment itself as a container in which the product and encompassing system of interest operate. To some degree, the testing environment is more complex than that product or system because it must include observability mechanisms for controlling the rate at which computation and inputs progress to permit in-progress examinations and analyses.

Beyond building the testing environment, there is the option of additional automation for the use of the environment. This includes the ability for one-step operation of scenarios that perform initialization, simulate operator inputs, record test results, and evaluate results against expectations. This is particularly useful for purposes of repeated regression testing as parts of the product evolve or to simulate and evaluate the product's behavior in response to potential future changes in its environment.

In the same way that the product evolves over its lifetime, the testing environment needs to accommodate changes in its representation of the operational environment. If the testing environment is built in a way that anticipates or is adaptable to future changes, it should be usable over the full lifetime of the product without significant additional investment. In fact, the testing environment should be viewed as an element of the product that is sustained in consonance with the product.

4 Evaluating a Product during Development

During product development, product evaluation is a collaboration between the developer and the acquirer. Product evaluation during development is meant to ensure that the resulting product will satisfy the customer's formal acceptance criteria with a minimum of evaluation effort during acceptance and without the need for rework.

4.1 The Product Development Process

Figure 2 depicts a generalized product development process. This process involves repeated feedback and iteration among the activities of development. Arrows indicate information flow between activities; activities may be performed simultaneously on different elements, increments, or versions of a developing product. This process is controlled by a continuous management activity that consists of planning and resource allocation, monitoring and metrics, configuration management, and process quality control. The nature of product evaluation during this process, including reviews and testing, is discussed in more detail below.

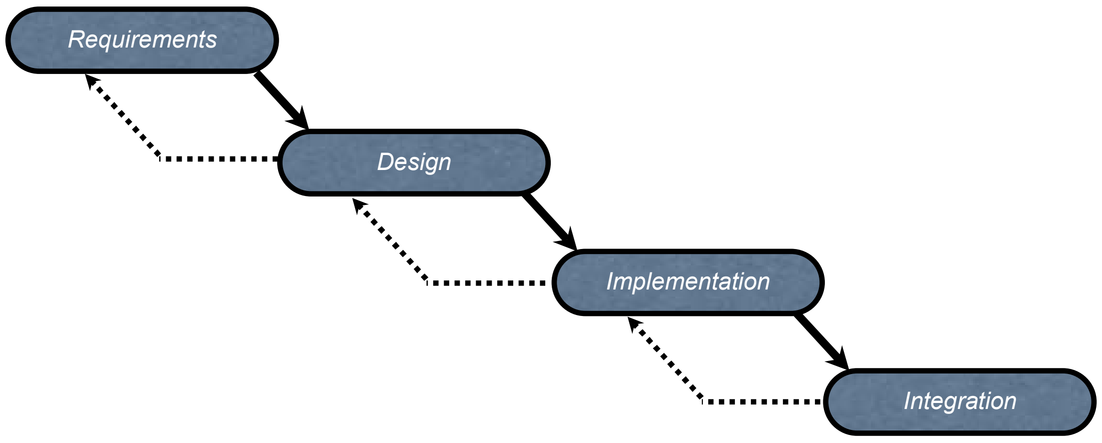

Figure 2: A Product Development Process

Although convention seems to favor distinguishing between activities of systems and software engineering, this discussion takes the view that software engineering activities are properly an aspect of systems engineering and cannot be deferred and treated separately [Campbell 2004]. Systems and software engineering alternatives and tradeoffs are interdependent; implications are often systemic, transcending any apparent distinctions. Differences in how hardware and software components are produced are the purview of the implementation activity; this facilitates deferring or changing the hardware/software implementation of any component. This means that other activities must address both system and software perspectives within the systems context: software requirements, design, and integration must be understood and addressed as an integral part of defining system requirements, design, and integration.

Product development begins with an analysis of acquirer-specified customer needs, resulting in the definition of acceptance criteria. These criteria are the formal basis against which product acceptance will be determined.

From these needs and criteria—augmented by engineering analyses of alternatives and tradeoffs—a specification of the product's requirements are derived. Requirements define the product's build-to observable behavior. As the product is developed (and later evolved), these requirements constitute an as-built specification of the product's expected behavior.

Verification that the product's requirements comply with acceptance criteria—along with subsequent verification that the product as built satisfies those requirements—should be sufficient to provide confidence that the product is going to satisfy product acceptance evaluations.

Product design specifies an architecture for the product, defining a consistent set of alternative views of the product's structure. These views specify the content, interdependencies, and interactions of the components that concretely implement the product. Further, each of these views constitutes a partial model of the product that provides answers to specific questions about how the product, and each of its components, is to be constructed. The following is one possible set of views:

- a module decomposition view that identifies and specifies the product's components
- a concurrency view that specifies the process/task scheduling and communications elements and interactions implemented by components
- a configuration view that specifies mapping of elements onto physical devices and computational hardware
- a dependency view that defines data and control dependencies among components and is used as a guide to the composition of operable product versions for incremental, reduced-capability integration

The design must be reviewed and analyzed to ensure that it documents and substantiates a technically credible solution approach that can be reasonably implemented with available time, expertise, and resources. The module decomposition view partitions the implementation effort into distinct components that can be assigned to developers based on their having requisite knowledge and expertise. All of the views defined in the design must be realized in the implementation of these components.

For each of the components specified in the design, an implementation is designed, written, and verified in keeping with the various design views. Implemented components should be verified through active reviews. Component-level testing can be used to analyze whether each component provides expected interface capabilities, behavior, and effects on quality factors. Testing can integrate already-verified dependent and supporting components to reduce the test preparation effort.

Integration, as the complement of design, creates an integrated operable version of the product by configuring and composing a selected set of verified components, as specified in the dependency view of the product architecture. This product version (which may be incomplete) is then verified against anticipated product behavior as specified by the requirements. The integrated product should be instrumented for use in the testing environment to enable dynamic monitoring, profiling, and analysis of its operational behavior. The product may be integrated with operational hardware and installed into its (actual or simulated) operational environment for more realistic, but controlled experimental use of the product following simplified and actual customer usage scenarios.

Product evaluation, as an element of integration, verifies that the product as built and operated within the development environment satisfies its requirements. If the requirements were verified as being responsive to customer acceptance criteria, it is reasonable to expect that the product should in turn satisfy the less restrictive customer acceptance criteria and therefore be ready for product acceptance. Defects that arise after product verification should be traceable predominantly to limitations in being able to replicate the system operational environment within the development environment.

4.2 The Role of Product Evaluation in Product Development

During product development, the role of product evaluation is to ensure that the completed product will satisfy the acquirer's and customer's acceptance criteria. Product evaluation therefore must participate in every product development activity.

In general, product evaluation's role is to determine whether

- information is presented in an understandable (clear, concise, consistent, well-structured) form
- terminology is properly defined and consistently used
- alternatives, tradeoffs, and rationale regarding important decisions are properly documented
- uncertainties, differing views, and potential future changes in needs, technology, or business context have been properly identified and addressed

For requirements, product evaluation's role is to determine whether

- customer needs have been adequately communicated by the acquirer, accurately documented in the acceptance criteria, and properly understood by the developers as expressed in the requirements
- requirements adequately define expected observable behavior for a product that, properly built, will satisfy specified acceptance criteria
- requirements specify that the product will include mechanisms and documentation of rationale as needed to make effective reviews and testing feasible
- changes are being made in requirements as development improves understanding of needs, alternatives, and tradeoffs to accurately represent the product being built

For design, product evaluation's role is to determine whether

- all aspects of requirements content are understood as communicated and represented adequately in the design
- different architectural views and models are defined as needed to permit analyses and evaluations of product behavior
- component specifications, including interfaces and dependencies, provide sufficient guidance to enable implementation by developers
- all significant quality factors have been identified and characterized in terms of design alternatives and tradeoffs

- the design provides means and mechanisms to operate properly in response to invalid actions or information, failures, and faults

For implementation, product evaluation's role is to determine whether

- the implications and realization of all architectural views for each component are understood by its implementer
- specifications of related components are being properly applied
- the internal design of each component identifies and properly organizes all of its elements
- each component properly implements its specification
- component verification criteria reflect the ways the component will be used

For integration, product evaluation's role is to determine whether

- the composition of a set of components, as specified in the design, is operable in the development test environment
- representative scenarios for expected use of the product behave as specified in the requirements

Throughout development, as well as during acceptance and sustainment, product evaluation must ensure that when a defect is discovered it is corrected and its cause is determined. Requisite corrections must then be propagated throughout all product representations to maintain consistency (e.g., correcting a design defect may also require changes in both requirements and implementation). Beyond this direct response, it should be determined whether an improvement in development or evaluation practices could preclude such defects in the future or cause them to be discovered earlier.

Product evaluation must determine whether a product satisfies customer acceptance criteria. Management may decide that a product is adequately responsive to customer needs and should be submitted for product acceptance even when certain criteria are not met. This can be the result of customer tradeoffs in cost schedule versus capability, better understanding of actual needs gained during development, or impending changes in customer needs. This judgment should result in the revision of the customer acceptance criteria with the reasons for this decision for reference during product acceptance and sustainment.

5 Evaluating a Product after Development

Regardless of whether an acquisition is considering an off-the-shelf (commercial or publicly obtainable) product or a custom-developed product, the product should be evaluated in terms of whether its observable behavior—augmented by documented evidence of its proper construction—satisfies identified customer needs. At this point, the acceptance criteria should be a clear representation of the customer's needs. This high level of confidence is due to the growing experience with the off-the-shelf product or to product evaluation efforts during product development.

To establish acceptability of a product, product evaluation is performed progressively to independently verify and validate the product:

- System verification: During development, the product has been verified, in a facsimile of the operational environment under representative operational scenarios, as meeting its requirements, and the requirements have been validated as consistent with customer acceptance criteria. The product must be integrated with other elements of the customer's operational system and verified as meeting aggregate customer acceptance criteria in a close approximation of the actual operational environment.
- Operational evaluation: having satisfied system-level product evaluation criteria, the product is subjected to validation involving representative operators/users in an accurate approximation of the operational environment and usage scenarios to determine if it is fit for its intended use.
- Delivery: A validated operational product must be installed with documentation, training, and assistance for users. Use of the product results in customer feedback on issues encountered, required workarounds, and projected future needs.
- Sustainment: An installed, in-use product over time begins to encounter changing customer needs. This leads to recurring performance of acquisition activities to produce, accept, and deliver revised versions of the product.

Finding defects either in product quality or in product fit to the customer's needs during product acceptance is costly, resulting in either a delay for rework or the acceptance of an inferior product. However, when a defect that prompts rework is discovered at this stage, action should be initiated not only to correct the defect through the required rework (depending on severity, immediately or in a future product revision), but also to determine why the defect was not discovered sooner, preferably during product development, and corrected then. A subsequent action should be to amend product evaluation procedures so that this and any similar defects are detected during development, rather than during product acceptance.

5.1 The Role of Product Evaluation in Determining Product Acceptance

Optimistically assuming that there are no defects, a product should conform to its associated as-built requirements. The requirements should define what behavior can be expected of the product in operation. The purpose of product evaluation in this case is to determine whether this expected behavior meets both customer acceptance criteria and more broadly, current actual customer needs. If the product fails to adequately satisfy this criteria, the acquirer and customer may decide to reject the product, accept the product, or consider critical changes to make the product acceptable. Changing a product that failed to satisfy its acceptance criteria requires making corresponding revisions in that criteria so that it matches. The evaluation of requirements during product development should have discovered any discrepancies with acceptance criteria; any failure to accept at this point would be the result of not having properly expressed the customer's needs or by an unanticipated change in the customer's needs.

Although requirements provide a nominal basis for evaluating the product, actual acceptance relies on criteria that may be implied in the requirements, but are made explicit only in the form of evaluation criteria. These criteria are created specifically as a guide for product evaluation, to define more precisely what requirements are understood to mean operationally.

A product is judged acceptable to the degree that it behaves in the way that the customer expects, regardless of whether the expected behavior was accurately communicated in the requirements. Product evaluation, as a proxy for acquirer and indirectly customer judgment, must establish through its realization of acceptance criteria whether a product is behaviorally acceptable for deployment and use.

5.2 Product Evaluation for the Purpose of Product Acceptance

When product development results in a candidate product release, the acquirer must determine whether the product as built is acceptable for deployment into operation.

Product acceptance includes three levels of product evaluation:

- Verification: determining that the product is a correct solution to the problem as understood and expressed in customer acceptance criteria.
- Validation: determining that the product properly addresses the customer's actual current and future needs.
- Certification: determining that the product satisfies all organizational and statutory requirements concerning safety, security, and legal regulation applicable to its operational environment.

Verification during product acceptance consists of reviews of documents for consistency and completeness and system testing. This is a determination that the product has been built as specified and works properly with other separately provided components of the system. If product evaluation was properly performed during development, failures of this type should not occur during acceptance; any verification defect found during product acceptance should be traced to its source in product development so that the product evaluation processes can be refined to expose this type of defect earlier.

Validation during product acceptance takes the form of operational/acceptance testing. This is the determination that the product provides the customer with the capabilities needed to perform their work effectively; this means that the requested capabilities correspond to what the customer actually needs at the time of completion. The evaluation can fail if the acquirer misunderstood the customer's needs or specified those needs incorrectly; this type of failure is less likely if representative users have been consulted during product development. Alternatively, this evaluation can fail if the developer misunderstood the customer's needs as communicated by the acquirer This misunderstanding should not be discovered during product acceptance, if during product development, the proper use of requirement and design reviews were conducted reflecting the customer's needs.

Validation requires the formal participation of customer representatives and is not constrained by formal acceptance criteria defined for the acquisition. The success of validation depends, however, on the acceptance criteria having accurately characterized the acquirer's and customer's expectations concerning the needed product's capabilities and behavior. If those needs have been misunderstood or misrepresented, or have subsequently changed, the product may be deemed unacceptable by the customer. The role of product evaluation in this respect is to establish and maintain visibility with the acquirer and customer representatives—and their active participation when possible—to ensure that the acceptance criteria properly describe their expectations. Acceptance criteria should include not only a characterization of customer needs, as they initially exist, but also a characterization of uncertainties and potential/anticipated changes in those needs. Awareness of uncertainties and prospective changes during development facilitates modifications in the product both before deployment and during sustainment.

Certification is the determination that the product conforms to organization, industry, and government rules and regulations regarding safety and security. The role of product evaluation in this process is to ensure that the product has been built in such a fashion that it will meet these criteria. This is substantiated by the proper recording, as part of process quality assurance, of associated safety and security assurance actions taken during development. As a rule, these criteria will be met if the product development process can be seen to have properly addressed the issues and tradeoffs that motivate standards of safety and security.

5.3 Product Evaluation as an Aspect of Sustainment

For most of their useful lives, products are in sustainment, encompassing both maintenance and follow-on acquisition. Maintenance includes monitoring and adjusting operational parameters, detecting and correcting residual defects, and replacing expendable parts and supplies, whereas follow-on acquisition involves modifying or replacing the product over time as the customer's needs change so that those needs will continue to be satisfied. As the proper complement to development, follow-on acquisition requires that all aspects of development and acceptance be repeated in order to ensure not only needed advances, but also continued integrity of the product's specification and behavior. Only by revisiting all aspects of development and acceptance can the acquirer and customer be assured that all implications of the required changes have been properly identified and addressed. Similarly, in the context of sustainment, all elements of product evaluation must be revisited to ensure proper repeat performance of an acquisition's product development and acceptance activities.

It is not sufficient during follow-on acquisition to consider only the direct effects of changes on the product—specifically for software but also to a degree for hardware. Implications of a change can often cascade into unexpected areas of behavior through complex low-level interactions such as indirect sharing of resources or unpredictable sequencing of communications and processing. In some cases, hardware can be sustained by substitution of one part with another of physically equivalent but improved capability; even in this case, as with software changes, improved capability usually implies potential behavioral differences that may change timing or informational dependencies with other parts of a system, requiring that the system be reevaluated.

As a sustainment activity, the details of product evaluation differ from prior occurrences to the degree that the product itself differs. There may be changes in requirements, design, implementation, and integration work products that require corresponding changes in the materials of product evaluation. For example, changes may mean that different quality factor tradeoffs would need to be considered or different user scenarios would need to be tested. Similarly, new, modified, or deleted components that integrate differently could result in new or changed product behavior and would require testing revisions. The infrastructure of product evaluation developed during the initial acquisition can still be used, but its details must be revised to account for the implications of any changes. During iterations of the initial acquisition, the product must be reevaluated to ensure that prior behavior remains unchanged with respect to any aspects that should not have been affected by the changes (i.e., regression reviews and testing).

6 Product Evaluation in a Product Line Context

The traditional, large system conception of acquisition is the development of a manufacturing facility that is operated to produce a series of largely identical products. More recently, the manufacturing approach has evolved to permit the products coming off the line to differ in ways that are more substantial. With the increasing reliance on software as a determinant of product capability, the opportunity exists to customize each product to satisfy specific operational needs. From a systems and software engineering perspective, this opportunity has been formalized as a systematic "product line" approach to acquisition [Campbell 2002].

6.1 The Concept of a Product Line

A product line is a set of products that satisfy similar needs. Differences in similar products may be essential—corresponding to differences in customers' needs—or they may be incidental, corresponding to differences in how developers have chosen to build the products.

Although similar products can and have been developed independently, doing so entails duplicate efforts during both development and sustainment and typically leads to unnecessary differences among those products. This duplication can be somewhat mitigated when the same engineers are responsible for those products, to the degree that they recognize and can maintain the similarity of the products over their lifetimes; however, a more systematic approach exists for creating coherent product lines. This approach—based on the concept of a product family—optimizes productivity and quality by minimizing essential differences and avoiding incidental differences among products.

A product family envisions a potential set of similar products; these products are alike in many ways (commonalities) but differ in certain particular ways (variabilities). A product family represents a set of products implicitly, in an abstract form that provides the means to rapidly derive a product that is customized to fit a customer's specific needs. Variabilities represent customer and engineering choices—in areas such as mission, business practices, technology, and operational environment—that must be made in order to build a specific product. These choices (in aggregate) determine the exact content of each work product, from requirements and design specifications to component implementations and user documentation. The products that are derived in this way, as instances of a product family, constitute a coherent product line. The effort to build individual products leverages the effort applied to creating the product family, resulting in the elimination of duplicate efforts across the product line.

An analogous case exists for treating a single product as a product line opportunity. For most products, there are uncertainties in customer needs and expectations for potential changes across the useful life of the product. These uncertainties and changes can be accommodated by conceiving of the evolving versions of the product as being distinct instances of an encompassing product line: each new version of the product could then be derived as an instance of that product line.

As discussed more broadly in McGregor, a product line approach suggests two opportunities for improving product evaluation costs and effectiveness. The first opportunity, exploiting the assumed similarity of products, leverages evaluation efforts across the product line by pre-verifying product line assets; this will improve the quality of each individual product while reducing the evaluation effort. The second opportunity concerns applying product line concepts in the development of product testing assets and capabilities [McGregor 2001].

6.2 Evaluating a Product Line to Reduce Product Evaluation Efforts

The first opportunity for improvement results directly from the need to build similar products. By specifying an architecture that describes any of the family of products equally well, a product line defines a set of components that can be reused to build customized instances of that product family. As with developing any component, assets should be separately evaluated prior to use in any product. These reviews and tests will provide an initial level of assurance that the components will be usable without further effort in any instance of the product family; in addition, because these assets are used in multiple products, subsequent testing of any instance product built using an asset has the effect to some degree of testing that asset across all similar products. With this added basis in prior testing, not only will evaluation efforts for future products be reduced, but those products will also exhibit higher quality, with fewer defects requiring less rework.

The objective with a product line is to substantially reduce the time required to create a product with a specified level of quality (e.g., no known critical defects and meeting threshold levels of all quality factors). Although a product line does enable the rapid, low-cost development of new and modified products, the artifacts that constitute product line assets should generally be evaluated prior to use against the same criteria and using the same techniques that would be used in evaluating a conventionally developed product. Even so, using properly evaluated product line assets can result in a substantial reduction in the time and expertise required to build and evaluate each subsequently derived instance product.

In a product line context, evaluations occur at the component, work product, and product levels. A product line asset at any of these levels corresponds to a set of similar instance assets; therefore, an evaluation may address either a single derived instance or a set of instances represented by the asset. Any asset evaluation consists of a combination of peer/mentor/expert reviews, testing, and formal methods. Any defect found in the specification or implementation of any asset instance is treated as being a defect in the asset as a whole.

A single-instance strategy for evaluating an asset entails (1) identifying relevant instance-level evaluation criteria, (2) deriving an instance that conforms to evaluation preconditions, (3) evaluating the instance relative to expected results, and (4) modifying the asset to correct for any defects. By selecting and evaluating a variety of instances of the asset, the quality of the asset as a whole will increase, along with confidence that any subsequent instance derived will be less likely to exhibit defects. This strategy is applicable when an instance for use in a product is derived and verified as part of the product evaluation effort; any defects that are found in this way should be used to correct the asset, not just the specific instance.

A refinement of a single-instance strategy is a sampled-instances strategy. By statistically selecting instances for evaluation, based on the distribution of references to the variabilities across the set, a higher level of confidence can be more systematically achieved for a given level of evaluation effort. By selecting instances that are sufficiently diverse to cover the set, less evaluation effort is needed to discover most asset defects.

The greatest leverage is provided by an abstract strategy for evaluating an asset. With this strategy, the abstract representation of the set of instances is directly analyzed. This requires using a method that permits reasoning logically over an abstractly characterized set. When using a formal method or model that was designed for verifying instance properties, variabilities correspond to universal quantifiers that extend the solution space over which the properties can be evaluated. This approach provides the means to demonstrate that a property holds—not just for individual asset instances—but also for every instance in the set. In addition, the conformance of the instance set to its specification can be reasoned over the logical space determined by the variabilities that characterize the asset as being a set.

In lieu of a formal method, evaluators can use review or static testing to analyze whether any combination of variability choices would result in an instance that would violate an asset's correctness properties. These informal techniques are imperfect but can increase confidence that all instances of the asset satisfy critical criteria without having to separately evaluate every derivable instance of the asset. These properties should still be verified specifically for each product that is subsequently derived for a customer; however, the effort to evaluate a derived product will be significantly reduced due to the improved quality, absence of defects, and reduced need for rework resulting from prior family-level evaluations.

6.3 Establishing a Testing Product Line

The second opportunity for improvement results from needing to repeatedly test multiple similar products or versions. Because products are similar, the required testing is also similar—requiring similar test plans, test scenarios and cases, test data, and test results—alike to the degree that customers' needs are alike and differing to the degree that they differ. By developing testing materials that are customizable in terms of how products' behaviors are meant to differ (i.e., treating them as product line assets), redundant preparations for separately testing each product can be eliminated. Investing in testing product line assets can pay off even if the tested products are being developed conventionally, as long as the similarity in their behaviors is understood.

Just as product line variabilities characterize customer and engineering choices that determine the exact content of each product, similar choices determine the content of each product's test materials. Rather than building test materials uniquely for each product or building them for one product and then modifying them uniquely for each succeeding product, product line testers would instead create adaptable test materials. Adaptable materials are explicitly customized, structured, and annotated based on the variabilities associated with the product line.

A further opportunity to leverage product line concepts for testing is to create an adaptable testing environment. With adaptability, the testing environment can be configured as needed to accurately replicate a variety of targeted business operational environments. This is beneficial when building similar products for different customers or for a customer whose operational environment can vary geographically or over time. In these cases, each product needs to be evaluated in a facsimile operational environment that has been tailored to match the customer's specified environment. The more sensitive a product's behavior is to its operational environment, the more important it is to evaluate the product in a test environment that accurately mimics the behavior of the actual environment. Again, rather than building or modifying a facsimile operational environment for every product, the facsimile environment can be constructed as an instance of a family that is to be adaptable to the variabilities that characterize the operational environments into which prod-ucts may be deployed. This can have the added advantage that the product will need less extensive testing in its actual operational environment, where the cost of testing as well as the risks and cost of correcting defects tend to be excessive.

7 Improving Product Evaluation Practices

Opportunities exist in most software development efforts to improve and streamline product evaluation practices. The preceding sections have provided some perspectives in general on how product evaluation can be performed most effectively within an acquisition effort. These perspectives were based on specific anecdotal experiences in working with a variety of DoD acquisition programs and their development contractors. Listed below are examples of limited propositions that may (in appropriate situations) result in beneficial improvements:

Establish precise, objective criteria for product acceptance

The role of product evaluation in product acceptance is to define a set of usage scenarios that will expose whether a product's behavior provides the capabilities expressed in customer needs. The basis for these scenarios should be acceptance criteria that formalize the acquirer's understanding of customer needs. Product acceptance should be defined as demonstrating that a product behaves as expected for a set of scenarios that properly represent the specified acceptance criteria. Any product that satisfies product requirements, which in turn has been verified as being in conformance with acceptance criteria, should operate properly under usage scenarios that are believed to be representative of criteria.

The goal of product development is to provide such a product. In addition, product quality should be measured in terms of the effectiveness of design tradeoffs in resolving conflicting quality attributes. Based on statistical analyses of product evaluation efforts prior to initiating product acceptance, the product should meet an expected level of confidence that it is free of unknown residual defects.

Designate product evaluators as authoritative proxies for the customer and associated users

The purpose of product evaluation is to ensure that the product as-built is the product that best meets the customer's actual needs. To achieve this, there must be a constant customer "presence" throughout development. Product evaluators may themselves be users but more typically will be people who broadly understand essential aspects of the customer enterprise, including users' roles, activities, and goals. This expertise should be augmented through direct liaison with other domain experts and knowledgeable users. Fixing a product that is defective but essentially fits the customer's needs is much easier than correcting a soundly built but conceptually flawed product.

Pair product evaluators with developers throughout product development

Product evaluators are likely to have a better understanding of what the customer needs as product developers work through alternative solutions. Product evaluators gain deeper insight into solution tradeoffs as product developers gain deeper insight into customer needs. In the case of component implementers, product evaluators will provide expertise and access to facilities that can reduce the developer's unit testing efforts and ensure readiness for integration testing. With appropriate expertise, a product evaluator can pair with developers of multiple components to gain improved design-level insights; similarly, developers who are familiar with the customer's needs can be product evaluators for other developers.

Perform product evaluation activities throughout product development

The lowest cost time to correct defects in a product is early, while the product is being developed. Reviews, model-based analyses, or tests, as appropriate to particular work products, should be performed continuously with a focus particularly on areas of greatest risk as judged by the developer. Requirements and design reviews should focus particularly on establishing bounds for systemic quality factors (e.g., performance, reliability, safety, security, usability) and how the product will satisfy these. Similarly, implementation reviews should focus on identifying and handling abnormal conditions.

Institute recurring active reviews of every work product as it is being developed

Focused peer/mentor/expert reviews are the earliest, least expensive, and most comprehensive means of finding defects in a product; active reviews are an effective method of performing such reviews. Reviewers are selected for specific expertise and focused on answering specific topical questions about aspects of the work product that the developer feels least certain of. An assigned product evaluator is a reviewer of first resort, with a focus on consistency, quality, and fit to customer needs. Reviews by authors of related development work products can expose inconsistencies and misunderstandings: in particular, requirements authors and designers should continuously review implementation work products to ensure developers have a shared understanding [Parnas 1985].

Substitute focused expert technical reviews for formal management progress reviews and technical oversight meetings

Management meetings typically consume too much time of too many people who should be doing other work. Automate tracking and reporting of planned progress, with management meetings scheduled “by exception” when advances or slips in schedules warrant re-planning or resolution of specific coordination issues. Use meetings as small, topically focused venues for resolving resource utilization conflicts or for identifying and resolving difficult requirements and engineering tradeoffs with responsible developers. Include knowledgeable evaluators and subject matter experts in these meetings.

Institute rapid iteration over the product development process

The best critics of a work product are other developers who use it: designers discover requirements issues, implementers discover design issues, and integrators discover implementation issues. Looking across all the work products, evaluators discover developer misconceptions, misunderstandings, and inconsistencies. No work product should be considered finished until it has been applied successfully, after correction of any discovered discrepancies. Incremental development and repeated iterative refinement of each work product is the shortest path to a quality product.

Task product evaluators to verify consistency across product development activities

Beyond establishing that each element of the product exhibits a correct understanding of customer needs, product evaluation needs to ensure that all elements are defining a consistent solution approach to meeting those needs.

Ensure that products are evaluated for proper behavior in nominal, stress, and abnormal conditions

Typically, the focus of evaluation is on how the product is expected to behave "normally," providing the capabilities that the customer needs; however, insufficient focus on product behavior under stress or abnormal conditions can result in missing defects that will lead to operational failures at the worst times. Stress conditions include performance and usability under heavy load such as communications or input data losses or reduced responsiveness to user actions. Abnormal conditions include device failures, data integrity losses under load, security lapses due to attempted misuse, or failures due to residual defects.

Recognize that an integration failure represents a lack of design-implementation discipline

It is the responsibility of the design to define the components that are implemented to build the product and how those components interact. Integration fails only if the design is flawed or a component implementation has failed to conform to the design. These failures should be found in design and implementation evaluations, not when integration is attempted—integration should be a purely mechanical process that never fails if implementation evaluations have been properly performed.

Plan to discover defects closest in time to their cause

The most cost-effective time to discover a defect is immediately after its creation. Deferral of efforts that could discover a defect until later in the process only serves to increase the difficulty of finding and correcting it and the risk that rework will take unplanned time and introduce new defects.

Whenever a defect is found, evaluate whether the defect could have been detected earlier. Identify product evaluation actions that would have detected such defects and plan suitable verification actions to ensure earlier detection of these defects in the future.

Use automation to reduce product evaluation effort

Product evaluation can be time consuming and repetitious. Such work can be automated, in part with commercial tools and otherwise using product line techniques. Initial setup of evaluation mechanisms may require manual effort but the application of these mechanisms and analysis of results against expectations should be largely automated. The goal should be to require human effort only by exception, such as when the product exhibits unexpected behavior. Recognize, however, that automation of product evaluation efforts is itself a development effort, requiring appropriate expertise in both software engineering methods and product evaluation techniques.

Breakdown artificial process boundaries

Treat product evaluation, including testing, not as a separate activity but as an integral aspect of every development activity. No activity should be considered complete until it has satisfied explicit evaluation criteria, with the goal of having no avoidable defects that will cause subsequent rework. Throughout development, developers and evaluators should collaboratively review and test the product to ensure with high certainty that post-development evaluation will not find any critical defects. Developers and evaluators should also work together throughout development to ensure that there is a common understanding of customer needs and tradeoffs so that defects will be avoided or found and corrected early.

Track effort in terms of tasks requiring similar skills rather than by project phase

Although a project might, for management purposes, be organized into a development and acceptance phase, both phases require performance of development and testing tasks. Staffing and budgets should reflect that these skills are needed in both phases. Rework that occurs during the test phase should be viewed as continuing development work, not as a normal aspect of testing. Similarly, product evaluators that help developers during the development phase to better test their work products are nevertheless performing test, not development, activities. Because of this functional overlap, improvements in either development or test practices will benefit both phases.

References

[Bennett 2005]
Bennett, Ted L. & Wennberg, Paul W. “Eliminating Embedded Software Defects Prior to Integration Test.” *CrossTalk* (December 2005): 13-18.

[Bhor 2001]
Bhor, Adrita. *Software Component Testing Strategies* (Technical Report UCI-ICS-02-06). Department of Information and Computer Science, University of California, Irvine, 2001.
www.ajevans.com/articles.php3?dlitem=componenttesting

[Boehm 2001]
Boehm, Barry & Basili, Victor R. “Software Defect Reduction Top 10 List.” *IEEE Computer* (January 2001): 135-137.

[Campbell 2002]
Campbell, Grady H., Jr. *A Software Product Line Vision for Defense Acquisition* (CMU/SEI-2002-TN-002). Software Engineering Institute, Carnegie Mellon University, 2002.
www.sei.cmu.edu/library/abstracts/reports/02tn002.cfm

[Campbell 2004]
Campbell, Grady. “Reconsidering the Role of Systems Engineering in DoD Software Problems.” Software Intensive Systems Acquisition Symposium, Software Engineering Institute, Carnegie Mellon University, 2004. www.sei.cmu.edu/library/abstracts/presentations/campbelljan2004.cfm

[Feiler 2010]
Feiler, Peter H. & Hansson, Jörgen. “Toward Model-Based Embedded System Validation through Virtual Integration.” *SoftwareTechNews 12*, 4 (January 2010): 26-33. softwaretech-news.thedacs.com/stn_view.php?stn_id=52&article_id=146

[Kopetz 1976]
Kopetz, H. *Software Reliability*. Springer-Verlag, 1976 (ISBN: 978-0333233726).

[Linger 1996]
Linger, Richard C. & Trammell, Carmen J. *Cleanroom Software Engineering Technology, Cleanroom Software Engineering Reference Model* (CMU/SEI-96-TR-022). Software Engineering Institute, Carnegie Mellon University, 1996.
www.sei.cmu.edu/library/abstracts/reports/96tr022.cfm

[McGregor 2001]
McGregor, John D. *Testing a Software Product Line* (CMU/SEI-2001-TR-022). Software Engineering Institute, Carnegie Mellon University, 2001.
www.sei.cmu.edu/library/abstracts/reports/01tr022.cfm

[Parnas 1985]
Parnas, David L. & Weiss, David M. “Active Design Reviews: Principles and Practices,” 215-222. *Proceedings of the 8th International Conference on Software Engineering*. Los Alamitos, CA, Aug. 1985. IEEE Computer Society Press, 1985.

[Perry 2000]
Perry, William E. *Effective Methods for Software Testing.* John Wiley & Sons, 2000 (ISBN: 978-0764598371). www.wiley.com/WileyCDA/WileyTitle/productCd-0764598376.html

[Pomeroy-Huff 2009]
Pomeroy-Huff, Marsha et al. “Competency Area 5: Planning and Tracking Software Quality,” 47-54. *The Personal Software Process (PSP) Body of Knowledge, Version 2.0* (CMU/SEI-2009-SR-018). Software Engineering Institute, Carnegie Mellon University
www.sei.cmu.edu/library/abstracts/reports/09sr018.cfm

[Spillner 2007]
Spillner, Andreas, Linz, Tilo, & Schaefer, Hans. *Software Testing Foundations.* Rocky Nook, 2007 (ISBN: 978-1933952086). www.rockynook.com/books/198.html

[Weinstock 2004]
Weinstock, Charles B., Goodenough, John B., & Hudak, John J. *Dependability Cases* (CMU/SEI-CMU-2004-TN-016). Software Engineering Institute, Carnegie Mellon University, 2004.
www.sei.cmu.edu/library/abstracts/reports/04tn016.cfm

REPORT DOCUMENTATION PAGE		*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE October 2011		3. REPORT TYPE AND DATES COVERED Final
4. TITLE AND SUBTITLE An Acquisition Perspective on Product Evaluation			5. FUNDING NUMBERS FA8721-05-C-0003
6. AUTHOR(S) Grady H. Campbell, Jr., Harry Levinson, and Richard Librizzi			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213			8. PERFORMING ORGANIZATION REPORT NUMBER CMU/SEI-2011-TN-007
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) HQ ESC/XPK 5 Eglin Street Hanscom AFB, MA 01731-2116			10. SPONSORING/MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES			
12A DISTRIBUTION/AVAILABILITY STATEMENT Unclassified/Unlimited, DTIC, NTIS			12B DISTRIBUTION CODE

13. ABSTRACT (MAXIMUM 200 WORDS)

This technical note focuses on software acquisition and development practices related to the evaluation of products before, during, and after implementation. From engagements with numerous DoD acquisition programs, it has been observed that a number of recurring issues reduce the effectiveness of how software-reliant products are evaluated. An acquisition effort consists of identifying the customer's needs, selecting or developing a product that is responsive to those needs, and then evaluating the product to determine if it properly addresses the identified needs. This technical note describes the Product Evaluation (verification, validation, and certification) process including test, reviews, and formal methods. It also makes the argument that Product Evaluation should not be deferred until after a product has been built, but should begin as soon as the customer's needs have been identified and should continue throughout the acquisition effort.

14. SUBJECT TERMS Acquisition, development practices, product evaluation, software-reliant products			15. NUMBER OF PAGES 42
16. PRICE CODE			
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL