2.8 Product Verification

The product verification model specifies substantiation that the product model as a whole is consistent and complete and that the product is a proper realization of the product model[1]. It identifies discrepancies for subsequent resolution among relevant elements of the product model. This model has three elements: static verifications of product model elements (e.g., reviews), analytic verifications of product qualities, and dynamic verifications of product behavior. These elements capture the evaluation criteria and results obtained using associated techniques.

Consistency is a pair-wise evaluation of all elements of the product model to identify differences due to changes and discrepancies between them that need to be reconciled. These evaluations occur when authors of related element specifications have baselined versions that are intended to be consistent.

Completeness is a determination that the product model is a sufficient basis for realizing a version of the product that can be empirically evaluated as to its value (utility and quality) in its actual ecosystem. Completeness is further a determination of the degree to which the product as a derivative of the product model provides a "true" realization of the capabilities specified in the product requirements model. This evaluation depends on the deployable product element of a baselined product delivery model and its associated baselined product requirements model.

The product verification model is the basis for determining whether a realized product version meets both customer acceptance criteria and project developmental quality criteria. To this end, the product requirements and product environment models must be consistent with the customer needs specification and all other elements of the product model must be mutually consistent with those.

Static Verifications Element

(coordinated reviews of all product elements)

---

[1] Based in part on ***An Acquisition Perspective on Product Evaluation*** (CMU/SEI-2011-TN-007), CMU Software Engineering Institute, Pittsburgh, PA, Oct 2011.

The static verifications element specifies the results of peer and expert reviews and static analysis tools in determining the consistency and completeness of all product model elements. A part of this specification is developed for each product model element. Each element is repeatedly reviewed based on its developer's need for evaluation of and alternative viewpoints about how uncertainties, inconsistencies, and tradeoffs within and among product model elements are most equitably resolved.

A review is a moderated interaction between the author of a model element and 2-4 designated reviewers to resolve author-specified concerns and identify additional issues related to the author-specified scope of the review. Additional reviewers may be given the opportunity to comment or identify issues in response to specific author-specified questions regarding a given element. The author designates reviewers as having relevant expertise or having responsibility for related elements. Each review follows a directed approach[2] as follows:

- A specific element of the product model, and any relevant dependencies with other elements, is the designated subject of a review.
- The characteristics of each reviewer involved in an element review.
- How the competence of each reviewer corresponds to the content of a reviewed element.
- The questions that are posed for each reviewer to answer based on their particular competence relative to specific aspects of the reviewed element.
- Specific assertions that each reviewer makes about a reviewed element.

Analytic Verifications Element

The analytic verifications element specifies the results of applying analytic evaluations to the product model or the behavior of the resulting product.

derived metrics (eg from testing-supplied measurements)

[2] Based on Parnas and Weiss, "Active Design Reviews: Principles and Practices"

predicted properties

comparative product realizations (evaluating relative product value for alternative product models)

Dynamic Verifications Element

The dynamic verifications element specifies the correspondence observed between the product model as specified and the behavior exhibited by a corresponding candidate product realization. This correspondence is the result of scenario-based operation of a product realization to obtain a comparison between expected and observed results for a specified testing event. The product model considered may be limited to any subset of the envisioned product (i.e., an assembly of related components) sufficient to perform a specified scenario.

Testing Platform

A testing platform is any mix as needed of all or parts of the actual or approximated product ecosystem, as specified in the product environment model, along with capabilities for instrumented monitoring and control of a testing event. Instrumentation includes the means to inject data and collect, retain, and report resulting computational results. A product is deployed into a testing platform in a form equivalent to that for its actual operational environment.

(creating a testing infrastructure; oper env defn; instrumentation mechanics; data collection-reporting; logging/data capture) (family of testing platforms or testing events?)

Testing Events

A testing event occurs on a testing platform with a subject of any component assembly. A component assembly is a subset of a baselined product instance, one or more components being verified along with any other supporting components needed to make the set as a whole operable within the testing platform. A product instance is derived as specified in the product delivery model, again to be operable within the product operational environment.

At all of these levels, the behaviors of the environment and any needed external elements may be simulated and any needed devices that are elements of the product may be emulated unless they are physically accessible. The product and any simulated or emulated elements may be instrumented to artificially control behavior and provide observability into the causes and effects of that behavior.

Verification involves three stages: preparation, execution, and analyses of results. Preparation is the plan and set up for a testing event. This entails an objective, customer-based or freeform scenario, a guide/playbook, testing platform set up including data initialization scripts, and expected results. Execution is the manual and/or automated performance record of a planned testing event (automated, directed, or device/user-role portrayal), including results of monitoring and control actions and collected data on behavior. Analyses of results correlates testing event occurrence to planned noting any divergences from expected with suspected cause and any remediative or diagnostic actions recommended for either the test event or its subject. Reporting should include results of any analytic evaluations of the testing event or resulting data.

Evaluation Criteria

The product requirements model defines the behavior that a resulting product is expected to exhibit. This model is defined to be consistent with the customer needs element of the product delivery model. The plan for a testing event will identify how its expected results are related to expected product behavior as whole, recognizing that a particular testing event may concern only a limited aspect of specified behavior.

Scenario Specifications

A primary basis for testing events is the set of operational scenarios defined in the product documentation element of the product delivery model. A scenario is specified in a form that provides sufficient information (preconditions, actions, and expected effects) to allow it to be performed (manually or automated as appropriate) as a testing event. A scenario must define initial conditions and data for testing platform initialization and correct observable behavior that should result. Results include any interactions that are expected to occur between the product and any entities in its

operating environment, as defined in the product environment model, and consequent effects on associated data. Scenarios should be conceived both to confirm occurrence of expected behavior and detect occurrence of unexpected behavior (e.g., due to invalid data values or as unforeseen effects of proper behavior).