2.7 Product Components

The product components model specifies the components, as defined in the product design model, that constitute the content with which a product is built. A component specification has three elements: its design (interface and internal specifications), its implementation, and its verification (substantiation that its implementation conforms to its design).

A software component is implemented in the form of a ***module***—a realization of the responsibilities assigned to that component by the product design[1]. Multiple discriminated modules can be associated with a component to provide alternative customized realizations of its capabilities. Each alternative must satisfy component responsibilities but can differ according to how it addresses tradeoffs and constraints on component capabilities and behavioral quality criteria. Additional modules can be created as needed to add, modify, or remove capabilities that distinguish them from other modules.

A dependent (i.e., client) component can selectively reference a specific module that best supports its needs. For example, there may be multiple modules implementing an "entity" component, ones that provide interactions with a physical realization of that entity and others that provide interactions with a virtual realization of that entity. Similarly, modules may differ in their tradeoffs among quality criteria (e.g., better performance versus higher data precision or stronger security), perhaps only partially supporting full component capabilities. Finally, two modules may differ only in whether some extrinsic capability, such as instrumentation that supports empirical evaluation, is included.

Component Design

The component design element elaborates the capabilities of a component, consistent with product requirements and in conformance with all elements of the product design.

---

[1] The ability to build components for reuse, particularly those having alternative implementations of component responsibilities (i.e., modules), are discussed in section 4.1.

Component capabilities include the behavior that the component enacts and any services it provides that support the implementation of other components.

A component design has two parts, an interface specification and an internal specification. These specifications describe the distinguishing capabilities of the component and its how associated modules differ in their realizations of component behavior, including functionality and quality.

Consideration must be given to whether and how a component's capabilities are likely to be changed, both during its initial development and over its useful life. Anticipating how a component is likely to evolve over time can result in a component design whose interface is less likely to require change and whose internals will require less effort to change as needed.

An interface specification establishes the conceptual integrity of the component. It describes, for its implementation and for client component implementations, a component's behavioral properties and its client-accessible services, consistent with its design-specified responsibilities, that other components can reference for their implementations. Behavioral properties include the functions it performs and the quality criteria that the component is expected to satisfy.

A component's interface specification defines an idealized view of the component's capabilities that can be assumed as fixed—unlikely to change. A component is implemented by one or more alternative modules that must conform to this specification. However, a module can restrict this specification both in terms of accessible capabilities and quality factors (e.g., providing different tradeoffs in conflicting quality criteria). Each module is characterized in terms of how it differs from others in both behavioral properties and provided services.

Each module is characterized in the component interface specification by how it specializes the component's capabilities as criteria for choosing among them for use by a client component. Any changes to an existing module's interface (e.g., to correct conceptual defects or deficiencies) may require changes to a dependent component's implementation. Changes to the component interface that require changing an existing

module interface are likely to propagate, forcing changes in dependent components as well.

{content of a component interface specification}

A component's internal specification describes the internal design and implementation of the component, identifying relevant references, assumptions, constraints, and differences or potential changes in needs or enabling technology. Alternatives and tradeoffs are identified as factors in determining what supporting services are needed to best implement each of the component's modules.

The factors that distinguish and constrain how each module's capabilities differ are described with associated rationale—why each module is needed as a specialization of component capabilities including to better satisfy particular client component needs—and any shared content or resources that modules should be implemented to use as needed.

{content of a component internal specification}

Component Implementation

The component implementation element consists of the implementation of one or more alternative modules that each provide a conforming specialized implementation of the component design. The internal design establishes the structure and form of each module.

Each module may exist in a series of versions, primarily during initial development and over time to correct defects. Conversely, the need for a module with more substantial differences—differing tradeoffs or constraints—may best be achieved with a new module, retaining existing alternatives as-is for continued use (e.g., to recreate previous versions of the product or being used as-is in other products).

A module may be implemented as a crafted artifact in a human-understandable "source" form[2] that is either "compiled" (mechanically transformed into a platform-executable "object" form) or "interpreted" (translated at runtime into an equivalent platform-supported computation). Alternatively, a module may be obtained from a separate, project-designated repository of previously built components or derived by means of an appropriately parameterized generative tool. In all cases, the module must be documented and verified according to relevant developmental quality criteria as specified by project management.

A module may be limited by the component design as to the other components or associated modules that it is allowed to reference for its implementation. A module obtained in an existing or derived source or object form (e.g., other separately developed software obtained from another provider having appropriate competence in needed capabilities or implemented as a framework in a runtime library of the computational platform) may need to be software-encapsulated so as to conform to the component interface specification.

Each module implementation should be augmented with associated information (e.g., comments) that (1) explains its primary constructs, assumptions, and dependencies and (2) elaborates the component internal design with rationale on alternatives considered and how constraints influenced that implementation, as may be relevant in making any future changes.

Component Verification

The component verification element specifies the means used to determine the degree to which the component implementation supports the responsibilities specified for it in the component design, including satisfaction of relevant product quality criteria and consistency with the interface specifications of referenced components. Means used to verify the component and each associated module include directed peer and expert

[2] A module in source form comprises programming language constructs, defining data constructs and processing logic (sequential and concurrent) intended to realize specified computational effects in software object form.

reviews, analytic evaluations, and limited usage-based testing (which may require implementing additional ancillary software that exercises each module's capabilities).

Criteria to be met by each module will differ according to how its realization of component capabilities is specified in the component design as differing. As a basis both for clients to choose among modules and for evaluating the effectiveness of the product design as a whole, this element should include an analytic evaluation of the relative importance of each of the quality factors, the degree to which each of the component's associated modules satisfy them, and any differences in tradeoffs among factors made in eacb module.