2.4 Product Requirements

The product requirements model specifies the behavior that a product is expected to exhibit, within a specified operational context[1,2]. This model is intended to establish a shared developer understanding of a problem-solution that is a consistent elaboration of customer needs as specified in the product delivery model. This model is augmented with a specification of envisioned product evolution over its useful life.

The product's behavior is defined in terms of its interactions with entities specified in the product environment model that constitute its operational environment. A product references entities both as sources of information and as agents for initiating actions that actualize the product's behavior.

During development, the requirements model is an evolving "build-to" specification of the product, defining all aspects of the observable behavior that the product is being built to exhibit. Upon deployment of the built product into operational use, the requirements model defines the correct "as-built" behavior of the product. The evolution of a product is viewed in terms of the aspects of its observable behavior that are expected to change over time.

Customer needs, as an expression of customer criteria for product acceptance, can be viewed as an under-constrained specification of the envisioned product in that many different products could be realized that would satisfy such criteria. Based on the flexibility that this provides for the developer to apply engineering judgement in identifying, exploring, and resolving uncertainties, alternatives, and tradeoffs, the product requirements model can be viewed as an over-constrained build-to/as-built specification, consistent with the customer needs specification but characterizing the observable behavior of a particular concretely-realized product.

Customer needs that are initially incomplete, not well understood, or poorly communicated as well as changes in developer understanding of the problem, solution

---

[1] K.L.Heninger, "Specifying Software Requirements for Complex Systems: New Techniques and Their Application", IEEE Trans on Sw Eng SE-6(1), Jan 1980.

[2] S. R. Faulk, "Understanding Software Requirements", 2011.

feasibility, or quality tradeoffs, and how the product can be built can lead to changes in the requirements model. The requirements model must necessarily also be modified if other elements of the project model cannot otherwise be realized in a way that is consistent with the requirements model.

This model specifies the product from 5 perspectives:

- Conception – a concise description of the product's purpose, in providing capabilities that support customer needs and objectives
- Information – a specification of the product's information content, on which its behavior is based and how this content is determined
- Behavior – a specification for the actions the product initiates through entities in its environment to induce effects on the ecosystem
- Quality – a characterization of the degree to which the product is expected to exhibit specified behavioral qualities
- Constraints – a characterization of extrinsic factors that limit the product's realization

Areas of Competence on Which Requirements is Based

<----------

For a proper conception of intended product behavior, the developer needs an understanding of customer needs: the targeted customer's objectives, how their enterprise operates, and how the envisioned product is expected to affect that operation. This understanding is based on two primary sources of insight:

- Business area competence – *knowledge* in the theory and practice of the type of enterprise targeted by the product; familiarity with *experiences* in similar and related operations and in developing similar and related products; and *expertise* with relevant techniques and technology appropriate for developing products of the sort needed

- Customer needs – a customer perspective on the purposes the product is expected to serve, how those purposes have been addressed previously, the capabilities that the product is expected to provide in support of its users, and any constraints that the product must satisfy

This model defines the "competence" of the product in exhibiting behavior: experience in this case is how envisioned behavior has previously been realized within this enterprise; expertise is the product's expected ability to enact reasoned/planned action; and knowledge is the product's awareness of the composition and functioning of the supported customer enterprise and ecosystem.

By applying business area competence to customer needs, the product requirements specifies the observable behavior of a specific product to be realized. Business area competence comes most easily from having worked in such an enterprise or having built products that solved similar problems, recognizing how differences in customers' circumstances can affect a solution. This understanding can be supplemented through active customer participation during development, or with program marketing as a customer proxy.

In developing a product through a series of increments, each increment typically targets only partial capabilities corresponding to specified customer needs. Subsequent increments add or revise capabilities over time to more closely match those needs. At the same time, actual or understood needs may change, resulting in changes in the product requirements model. Increments continue until a version is completed that is acceptable for delivery. A delivered version may still lack capabilities, deferred with customer concurrence due to cost-schedule constraints to a future release.

Customer needs are sometimes inadvertently constraining, due to omissions, uncertainties, ambiguities, or nonessential constraints. These issues can lead to excessive solution complexity, portending delay, increased cost, or infeasibility. The developer may suggest revisions in customer needs that will mitigate or defer such complexity, allowing the customer to influence the tradeoff between needs and effort. A feasible option is to develop an interim more limited version of the product that can be revised in subsequent increments as these issues are resolved.

---------->

Specifying Product Conception

The product conception element is the product "vision" from a developer perspective. It concisely specifies, for a realization-independent shared understanding by developers, the purpose of the product from a customer perspective. This includes the nature of the operational context in which it is used, the type of information that it manages, and the capabilities that it provides within the customer enterprise.

This element is meant to suffice in describing any realizable product that would satisfy a customer's current and likely future needs. This element is augmented with definitions of key terminology used and references to materials that further explain the nature of the problem-solution space that it addresses.

Specifying Product Information

{product information can be represented with an active semantic data model; this includes (1) information defined in the product environment model (from physically or virtually-realized entities, including the environment as a "virtual" entity) and (2) computationally-derived (inferred) content}

The product information element specifies the information content on which product behavior is based. Ecosystem information (native environment or entity-associated content) is obtained via interactions with entities operating in the environment. Primary content traces to information content specified by the product environment model. Additional content can be specified that is computationally derivative of other content or that models the effects of past or potential future ecosystem or product behavior.

This element defines all information that the product references in order to determine its behavior within its operational environment. This information depicts the changing state (past, present, and future, as appropriate) of the ecosystem upon which product behavior is based. Changes in ecosystem state are obtained from entities operating in the environment or inferred based on logical dependencies among related information.

Specifying Product Behavior

{alternative forms of specifying "behavior" (include modes/user-roles, conditional event-/demand-driven behavior of each entity) in which elements can differ depending on the nature of the product being developed and entity capabilities; characterizations here can only give a general sense of the form that such specifications can take}

The product behavior element specifies how the product interacts with accessible entities to initiate observable effects on its operational environment. Each effect is specified as a function that, based on changes in product information content, requests an entity action that is intended to influence ecosystem information content[3]. The behavior element has two parts: activation-continuation-termination criteria, and action content.

The product environment model specifies the entities that are accessible to the product and the set of addressable actions associated with each entity. Behavior is expressed in a formalism that is characteristic of the sorts of interactions that a given type of entity accommodates (i.e., the actions that an entity recognizes as defined in the product environment model). An action is the conditional dispatch of a request and associated data to a designated entity and service. Each action is initiated periodically or on demand due to changes in product information content (due to detection ecosystem phenomena or effects of other product behavior).

Activation, Continuation, and Termination Criteria

Activation criteria is a specification of the operational condition that causes an action to be initiated. Actions can be initiated based on one or any combination of (1) a specific request (i.e., event) initiated by another action, (2) a transition in the value of a predicate concerning the content of monitored data, or (3) a transition in operational mode of the product instance. When given criteria is satisfied, the associated action is initiated according to its specified effects as appropriate to the type of device it targets.

Continuation criteria ….

Termination criteria ….

[3] S. Faulk, et al, "The Core Method for Real-Time Requirements", IEEE Software, Sep 1992, 22-33.

Completion of an action may initiate other actions depending on the results, including reacting to failure of its action, propagating data that will result in changes in information, or triggering other dependent behavior.

Actions

Each entity supports a specified set of actions. An action can be a request to modify environment information or to coordinate a dialog. Edge devices interact directly with the product's native (physical or virtualized) environment to obtain or change associated information content. Interface devices interact with users or systems to share information or coordinate actions.

Edge device actions support actions, for devices interacting directly with the product's native environment, that initiate changes in the environment as needed to realize some intended product behavior.

Interface device actions initiate actions for coordinating actions or sharing information with user or system entities operating in the same operational environment. Each such action is defined in a form that is known and can be enacted on designated entities.

User interface actions initiate actions via a virtual interface device that supports interactions via one or more logical interface devices with operators and users of product capabilities. A user device is characterized as supporting a specific user/ operator "role". Each role is specified as allowing a coherent set of responsibilities in enterprise operations. A user device presents data to convey information within the purview and operational context of users performing the activities associated with a designated role. Data may have associated mechanisms associated with a given type of user interface device for responding to user actions. *{specify the forms of display and interaction that a user device can support conceptually[4] here and logically in design model}*

System interface actions initiate actions via an interface device that transfers action requests or data to one or more specified interface devices defined as associated with a

[4] G. Campbell, "An Approach to Specifying Requirements for User Interfaces", Software Cost Reduction Workshop, Nov 1994.

specified type of system entity. Each system interface device is defined in terms of the form in which understood actions and data can be accepted.

Specifying Product Quality

The product quality element specifies the criteria by which the product is built and evaluated as acceptable for its intended use. Product quality is defined in terms of the acceptable limits on various properties that determine the fitness of a product for its intended purpose. This element defines which specific properties apply to the product and provides guidance on making tradeoffs among the properties to resolve conflicts among properties so as to achieve a viable product realization.

Product quality is expressed as "behavioral" quality—the degree to which each aspect of quality is to be exhibited in a product's observable behavior. Behavioral quality can be expressed in four broad facets: functionality, performance, dependability, and usability. While all four of these facets are relevant for any product, the specific properties encompassed by each and the degree of influence that each property has on acceptability of the product can differ based on the nature of the product and the circumstances of its intended use. {A notional formulation of the properties in each behavioral quality facet is given below.}

The relative importance of each property to the product as a whole is characterized in the product quality element. Each property should be characterized in terms of how it is to be evaluated as a factor in acceptable product behavior. Some properties will be expected to exhibit an objective measure, predictably achieving either a minimum acceptable value or a nominal range-limited value, within indicated tolerances. Other properties may only be subjectively evaluated through reviews based on customer criteria.

This provides guidance as to how each property is expected to influence the product design including how tradeoffs among related properties can be resolved. The product design model substantiates whether the specified quality factor goals, individually and in aggregate, are satisfiable as a design emerges, both during initial development and as the product is subsequently modified. Design alternatives are considered in terms of

how well quality criteria is satisfied by each. Alternatives may be motivated by exploring different tradeoffs in how quality factors are affected. If a viable design cannot be determined based on provided guidance, it may be necessary to relax this guidance.

The interface design element for each design-specified component may be required to specify the relevance of each behavioral property in its realization. The design model is expected to substantiate that each property is properly achieved by the combination of components that influence that property.

{rqmts specifies quality criteria in terms of observable behavior of the product as a whole but significance of any factor can differ for each element of product information or product behavior} {2.5 must address how quality factor goals are differently allocated to/across different parts of a product (e.g., a factor such as safety will be directly addressed only within some portions of the product architecture; things such as the frequency at which an output must occur or the precision of a data item can be specified for each element of information or behavior as appropriate)}

Notional Behavioral Qualities (1)

Functionality, the degree to which a product exhibits expected behavior (i.e., effectiveness); typical elements include:

- *Utility* (satisfies its intended purpose including aligning with and supporting encompassing enterprise operations)
- *Interoperability* (operates and communicates properly with entities in its ecosystem)
- *Adaptivity* (has the means to modify its own behavior as directed (i.e., reconfigurability for e.g., localization, specialization, or personalization) or in response to prescribed circumstances such as high demand, fault, or degraded conditions)

Performance, the degree to which a product supports an anticipated workload, consistent with available resource capacities; typical elements include:

- *Responsiveness* (reacts to external stimuli and internal effects sufficiently to maintain consistency internally and with respect to its ecosystem)
- *Efficiency* (minimizes use of available physical and logistical resources, including utilization of energy and impacts on environmental conditions) {conservance}
- *Throughput* (produces effects at a rate sufficient to maintain consistency with external elements of the environment and ecosystem)
- *Scalability* (adjusts to projected variations in workload and resource capacities)

Notional Behavioral Qualities (2)

Dependability, the degree to which a product continues to produce expected effects (behavior and data) under all (normal, abnormal, and unforeseen) conditions; typical elements include:

- *Reproducibility* (exhibits differences in behavior or information content only as a result of differences in external influences)
- *Reliability* (exhibits correct and consistent effects under normal and degraded conditions) {includes predictability, stability, and survivability, including resilience and recoverability}
- *Availability* (operates without interruption or unavailability of information) {includes continuity and connectivity}
- *Integrity* (provides valid and complete information, while excluding unauthorized access and precluding unspecified behavior) {includes privacy/security and accuracy and precision of information (ameliorating false, biased, or unreliable content)}
- *Safety* (prevents or detects and mitigates conditions, actions, or information that would cause damage to any part of itself or its ecosystem)

Usability, the degree to which a product is able to be used properly, given the responsibilities of its intended users/operators; typical elements include:

- *Conformability* (operates consistent with legal, financial, ethical, and equity dictates and the competencies (language-terminology, knowledge-expertise, and skills-abilities) of users) {includes accessibility}
- *Learnability* (aids and instructs users in the nature and use of its capabilities and information content)
- *Explainability* (is able to communicate causes and rationale for prior or prospective behavior and information content) {includes transparency and accountability}
- *Aesthetics* (presents and properly conveys provided information and functionality in a form that facilitates intended usage) {includes consistency}

Specifying Product Constraints

The product constraints element specifies extrinsic factors that limit options for the product's realization. These factors can reflect provider- or customer-instituted policies and practices, industry standards, or government-mandated regulations. Constraints that cannot be relaxed may preclude alternative solutions that might be preferred if left only to engineering judgement.

Project management, in consultation with program management, must resolve any conflicts among these constraints or any constraints that conflict with aspects of envisioned product behavior. Any potential changes in applicable constraints should be considered in building the product for ease of future changes that may be needed to suit evolving customer circumstances.

External Constraints

Constraints may be imposed by external authority. Formal authority includes the various governmental legal and regulatory jurisdictions that establish standards for accountability, infrastructure, communication, safety, security, data privacy, and environmental impacts. Relevant industry and market governing bodies may establish additional mandatory or discretionary technical standards and conventions to be followed. Compliance with these constraints my limit other choices such as tools and operating software to be used.

Customer-imposed Constraints

The product must be built to conform to constraints associated with the customer's business and intended operational environment(s), fitting within and supporting customer operations. The customer relationship element of the project management model and the customer needs element of the product delivery model may identify (or imply) constraints arising from customer enterprise or operational conventions.

The customer may designate interface conventions associated with tools and technologies or systems that reflect the enterprise's envisioned use of the product. These may entail expected use of the customer's computational environment, particular commercial tools or devices, data security protocols, monitoring and auditing practices,

or other organizational conventions that the product must satisfy. These constraints may affect the platform on which the product is built to operate. (The customer may also impose constraints on developmental practices that project management directs be used in building the product.)

Examples of constraints that could be imposed include:

- Compliance with externally-defined or similar customer-instituted standards
- Designated use of existing or planned customer facilities or preferred commercially-available ("COTS") products (including tools, data storage, communications protocols, or computational platform)
- User interface conventions (i.e., the forms in which information is presented to users according to role)
- Computational platform standards and conventions regarding computation and data distribution, resource usage, messaging, transaction protocols, and logging and auditing practices
- Platform management protocols (startup/shutdown, hardware health and diagnostic conventions, concurrency techniques, fault handling and degraded processing)
- Platform integrity (system, transactional, and data access security and safety practices and other related dependability quality criteria)
- Hardware-software configuration, compatibility, and reconfiguration/reprogramming practices
- Deployment practices (installation, training, and logistics)
- Accommodations for operational and computational environment evolution, considering requirements if any for continuous operation

Program-imposed Constraints

Program management, consistent with enterprise guidance, may impose program-beneficial constraints that limit both developer and customer options for conventions

followed in defining product behavior. These constraints focus on avoiding allocation of program resources and efforts that are not conducive to overall productivity in achieving long-term program objectives. This includes conventions for consistency in the formulation of common product capabilities, for best use of shared assets in a product's behavior and how it is allowed to interact with instances of related products and with relevant entities in its operational environment. Interactions may be constrained as to the entities and products to be referenced and the protocols to be used in communicating with these.

Product Evolution

The product evolution specification characterizes how required the product is expected to change over time, as a guide to what changes to the product model as a whole are likely to be needed. Developers in building the product are expected to consider how such changes will be accommodated as they arise without undue effort. Such changes will be specified as part of major product releases in the product master plan as they occur.

Product evolution can be understood in terms of the concept of product subsets. Each version of the product that is needed at a given time can be thought of as a subset of an envisioned maximum realization of the product that includes the set of all capabilities (including alternate versions of any particular capability) that will ever be needed over the useful life of the evolving product. A given version is derived by removing those capabilities that are not needed for that particular version. The maximum realization never exists as such but is the combination of all efforts over the entirety a product's development. The purpose of the product evolution specification is to envision all the versions that are going to be needed so that, as changes occur, the changes in the product at that time are a reasonable fit, requiring no more effort than if they had all been built in the first version of the product.

Actual product development can mimic this expectation by organizing product model content to include informative annotations and placeholder code that anticipate how the existing content would need to be changed if some aspect of expected product evolution is realized at some future time. In some cases, potential changes may

correspond to explorations of alternative solutions which can be retained as an option that will support those changes. As an envisioned progressive series of future modifications are made, these annotations would need to be modified in anticipation of other projected future changes.

As with initial product development, product evolution efforts may be limited by availability of resources and quality tradeoffs that limit feasibility or viability of needed changes. Developing with the intent of making future changes less costly can mitigate these limitations.

Product evolution is envisioned in terms of accommodating expected changes over time in customer needs and in enabling technology. Such changes equate to changes over time in either the problem or its solution. These are accommodated by corresponding changes in the product model. Even when the specific details of future changes may not be known, the recognition of the aspects of problem or solution that may be likely to change can be the basis for organizing content so that likely changes are easier than unlikely changes. The corresponding types of changes that drive product evolution include:

- How the operational environment, including capabilities of associated types of entities, is likely to change over time
- How customer operations are likely to change over time, including changes in operational capabilities that the product needs to support
- How externally-imposed constraints are likely to change over time
- How enabling technology, including COTS products, are likely to change
- How quality criteria is likely to change

This specification may be modified as unforeseen changes in any of the above areas occur, corresponding to different assumptions concerning how the product is expected to evolve over time.