2.4 Product Requirements

The product requirements model specifies the behavior that a product is expected to exhibit, within a specified operational context.[1] 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 expected product evolution over its envisioned useful life.

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

Customer needs, as an expression of customer criteria for product acceptance, should 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, ambiguities, alternatives, and tradeoffs, the product requirements model is meant to be an over-constrained specification, consistent with the customer needs specification but characterizing the observable behavior of a particular concretely-realized product.

During development, the requirements model is an evolving "build-to" specification of the product, aiming to define all aspects of the observable behavior that the product is meant to exhibit. Changes in developer understanding of the problem, solution feasibility, quality tradeoffs, or constraints can lead to changes in the customer needs model. Similarly changes in the customer needs model may lead to changes in the requirements model. Upon completion and deployment of the realized 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 may change over time.

---

[1] S. R. Faulk, "Understanding Software Requirements", 2011. < https://classes.cs.uoregon.edu/17W/cis422/readings/Faulk_SoftwareRequirements.pdf >

A developer familiar with the customer's subject matter, such as from having worked previously in building similar or related products, is best prepared to formulate requirements that align with the customer's actual needs. A developer having such familiarity will require less time to understand the customer's perspective as expressed in customer needs—use of specialized terminology, the purpose and needed capabilities of the envisioned product, and how that purpose is being addressed currently. This familiarity will give the developer credibility in suggesting changes to achieve product feasibility, while allowing the customer to influence the tradeoff between needs and timely product availability. This credibility will be enhanced if the developer recognizes how differences in customer circumstances can warrant differing solutions. This effort should be supported through active customer participation during development (or as needed for a simple market, with program marketing as a customer proxy).

This model specifies the product from 4 perspectives:

- Concept – a concise description of the product's purpose, in providing capabilities that support customer needs and objectives
- Behavior – a specification for the actions that the product initiates through interactions with 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

Product Concept

The product concept 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, in conjunction with the product environment model, 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, with anticipated changes, likely future needs. This element is augmented with definitions of key terminology and references to materials that further explain the nature of the problem-solution space that it addresses.

Product Behavior

The product behavior element specifies how the product interacts with accessible entities to monitor relevant information and effect behavior within its operational environment. The product environment model specifies the entities that are accessible to the product and services associated with each entity as appropriate to its type (edge device, user role, or aggregate/system).

Product behavior is determined based on entity-reported and product-derived data. An entity may have the means to conditionally share relevant ecosystem content with the product: as changes in that content are detected, on a periodic schedule, or upon request by the product. Provided information may have associated metadata such as when and on what basis its value has been determined.

[This section provides a broad characterization of the expected content of a behavior specification without imposing a specific formulation as would be determined by a particular requirements method[2].]

Product behavior is expressed as a set of *activities*. Each activity is either composite or entity-specific. Each activity has associated activation criteria (to become enabled); it may have continuation criteria (for repetition while enabled) and termination criteria (to become disabled). It may have an associated follow-up/continuation action if needed (e.g., acting on success or failure of the action).

A composite activity provides for the coordinated management of logically related activities (e.g., defining a "dialog" or result-dependent action such as acting on an interrupted or failed request, disseminating data among related activities, or initiating dependent behavior). Composition may specify conditional initiation of multiple

[2] As one example, see the "behavioral model" content in S. Faulk, et al, *Consortium Requirement Engineering Guidebook (SPC-92060-CMC)*, Dec 1993. < https://apps.dtic.mil/sti/pdfs/ADA274691.pdf >

activities—a sequencing (inclusion or delegation of responsibility), a selection, or a concurrent set of activities.

An entity-specific activity specifies actions that it can request to be initiated by a corresponding entity. Each entity is specified in the environment model as to the services it provides that may be needed by a product. The intended effect of each entity-supported service is to modify its own or other ecosystem information content (including by initiating entity-accessible physical processes).

An activity consists of a set of mutually exclusive actions, each action specifying its criteria for initiating a request and the information to be provided with it. (An entity may be replicated within an ecosystem in which case an action applies to all accessible instances, unless selectively restricted according to entity-distinguishing properties, such as identity, location, or operating status).

The nature of services provided by an entity may differ depending on whether it represents an edge device, a user role, or an aggregate/system. An edge entity supports actions that directly act on elements of the native (physical or virtualized) ecosystem. A user entity supports actions[3] by which authorized users, interacting via platform interface devices in performing their assigned activities, can obtain or report relevant ecosystem information and request product-supported actions including delegation of consequent actions. An aggregate entity supports actions, as designated for it, to share information or coordinate activity with the product via platform interface devices.

<----------

Conditions are predicates on ecosystem (environment and entity) information and product operational status. Operational status is an abbreviation of the history of product operation, for example, whether particular activities have been activated. Predicates corresponding to operational status, based on a product's functionality, may be condensed to a set of *modes*, such as "operating", "unavailable", or "constrained". A mode may be partitioned to distinguish operational limitations. An entity may similarly

[3] G. Campbell, "An Approach to Specifying Requirements for User Interfaces", Software Cost Reduction Workshop, Naval Research Laboratory, Washington, DC, Nov 1994.

expose its status in terms of operating modes. A set of modes representing the operational status of each entity can be similarly characterized and referenced. The concept of authorization (e.g., of a user or system entity to access particular information or initiate particular activities) applies similarly to modes.

---------->

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's observable behavior as a whole and provides guidance on making tradeoffs to resolve any conflicts among those 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.}

{identify elements of product behavior that are most significant determinants in satisfying various quality criteria}

The relative importance of each property to the product as a whole is characterized in this 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 of behavior based on customer criteria.

This element 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 each satisfies quality criteria, exploring 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 design model is expected to substantiate that each property is properly achieved by the combination of components that influence that property. To this end, the interface design element for each design-specified component may be required to specify the relevance of each behavioral property in its realization. Quality factor goals may be differently allocated across components of the product architecture (e.g., a factor such as safety, data integrity, or responsiveness will be directly addressed only within some portions of the product architecture).

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:

- *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) {includes redundancy and fault tolerance}

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}

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 may 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 (e.g., contractually imposed) 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 analogous 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- or Project-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. The program or project may impose additional constraints to ensure consistency across related projects and products of the provider or customer enterprise. 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 the product is expected to change over its useful life, 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 without undue effort as they arise. Such changes will be anticipated in the product master plan.

Product evolution can be understood in terms of the concept of product subsets. Each version of the product as needed at a given time can be thought of as having a subset of all buildable capabilities that will ever be needed over the useful life of the evolving product. A given version omits those capabilities that are not needed for that version.

The purpose of the product evolution specification is to envision a series of versions of the product 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 been built in the first version of the product. Potential changes are not intended to be exhaustively identified but rather only considered to the degree necessary to have a general idea what aspects of the product are likely to be changed over time and some sense of how; this is meant to influence developers as they build preceding versions to

consider how likely changes will need to be applied to the version they are building, making choices that will make changes less disruptive and costly than if they had not been considered.

Developers can pursue 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 when some aspect of expected product evolution is later realized. In some cases, routine explorations of alternative solutions can be retained as an alternative solution that would satisfy a future change. 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 enabling technology. Such changes equate to changes in either the problem or its solution. These are accommodated by corresponding changes in the product model. Even though the specific details of future changes may not be known, the recognition of the aspects of the problem or solution that may be likely to change can be the basis for organizing content so that likely changes are less costly and easier to make than unexpected changes that may arise. 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, is likely to change over time

- How quality criteria is likely to change over time

This specification should be modified as changes, foreseen or not, progress in any of the above areas, corresponding to different assumptions concerning how the product is likely to evolve after that time.