2.1 The Context for Software Product Engineering

The context in which software product engineering is performed concerns its position in the organizational structure of its parent enterprise, its relation to any non-software aspects of providing its product to customers, and the perspective its customers have about its product.

The organizational structure of an enterprise can differ depending on its objectives. The simple model described here consists of executive, program, and project levels. While an enterprise could have a more complex management structure, this structure is sufficient to illustrate how a software production capability can be effectively positioned within an enterprise.

The executive level is responsible for enterprise governance, defining the objectives and administration of the enterprise. The executive level charters a set of programs, each with a specified scope of activity, including targeted market and technical competence, which in aggregate is meant to achieve the objectives of the enterprise. Each program then initiates projects, each chartered to be responsible for a specific customer-targeted product. A project chartered as a product provider would be responsible for creating and maintaining an ongoing relationship with its (internal or external) customer/simple market to create and evolve a product that meets their needs.

In more detail, there are six elements that provide the context in which software product engineering operates; each of these establishes conditions that frame a software project and its resulting product:

- The nature of enterprise governance and executive management under which the effort is chartered
- The nature of responsible program management with a designated market focus
- Data as an approximation of reality
- Hardware engineering for a product platform and associated devices
- Systems engineering for the broader system-level context of a product and its dependencies on related entities

- Customer accommodation and Product lifecycle implications for customers: The concept of acquisition as a customer-side complement to product providers, organizational transition, and customer implications of product evolution

Enterprise Governance and Executive Management

An enterprise is initiated by one or more chartering entities, from which it obtains its mission, core resources, and imposed conditions on its operation. The enterprise operates under an executive management that defines mission-directed objectives and strategy, obtains additional resources (funding, people, facilities), imposes uniform practices and criteria, and evaluates and reports on progress. Executive management charters programs to operate in a designated area of endeavor related to enterprise objectives, delegating appropriate mission and operational objectives, resources, and conditions to each.

The enterprise supports programs by instituting common administrative services, including legal, contracting, and procurement, finance and accounting, human resources, facilities and information technology, and logistics. Executive management may institute enterprise-wide standards (policies, procedures, conventions, and practices) as appropriate including constraints on organizational and technological options as needed to ensure consistency or conservation of resources across programs. Enterprise-imposed standards may evolve based on feedback from programs regarding issues and changing needs. If programs are found to have common or related interests, management may guide those programs in collaborating to address those interests, seeking cost-benefit advantages for the enterprise as a whole.

Program Management

A program is chartered by executive management with responsibility for pursing specified objectives and is managed in accordance with the guidance, coordination, and oversight of executive management. Program management is responsible for determining how to achieve those objectives with allocated resources. A program may be initiated to address a designated market corresponding to a (possibly changing) set of potential customers that may benefit from using products and services that the

organization has the competence to provide. The objective for a market-directed program is to coevolve with the market, creating and sustaining products that will address the market's changing needs and, by means of those products, influencing the ways in which market and customer needs are likely to change. To the degree that customer needs are similar, program management may seek to leverage work across projects.

Each program initiates a marketing function to identify and understand the types of needs that the program has been created to address. Marketing has responsibility for characterizing current and future market needs as they relate to the program's charter, making customers aware of program capabilities as those evolve, and identifying and initiating relationships with potential customers. It further is responsible for characterizing the capabilities that products should have to be responsive to customer needs.

A program serves its targeted market through the auspices of chartered projects [Figure 2.1-1]. Each project is initiated to address the needs of a designated customer (i.e., either a single customer having unique needs or a simple market consisting of customers whose needs are sufficiently similar to be satisfied by a single product). (A project's customer may consist of other projects within the same program or enterprise.)

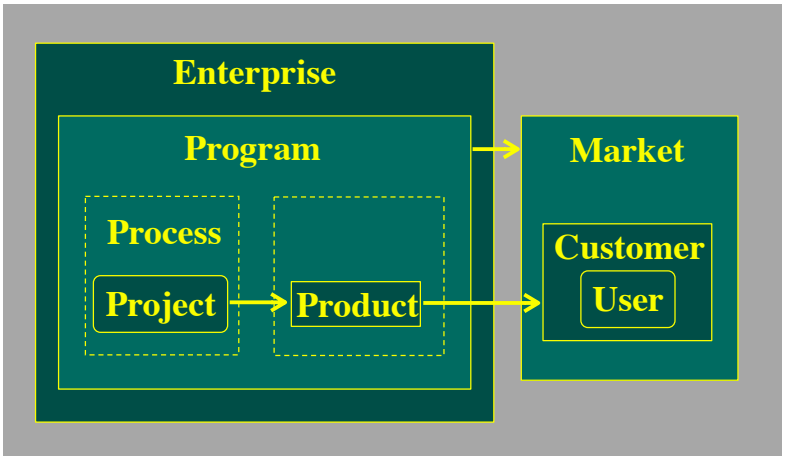

Figure 2.1-1. Program-Market and Project-Customer Relationships

Program marketing is responsible to each project for facilitating a relationship with its designated customer and for providing assistance in understanding that customer's

current and future needs in relation to the program's market as whole. If a project's customer is a simple market, the program may establish a marketing-adjunct sales function focused on engaging new customers for the product targeting that market.

Each project works with its designated customer to establish a collaborative relationship to best address their needs given program capabilities. A project is meant to operate as long as it continues to perform in accordance with its defined objective. Each provider project is responsible for developing, supporting, and evolving a product, consistent with program capabilities, that will address its customer's specific needs.

Program management coordinates with enterprise administrative services to ensure that projects have the means to operate efficiently and effectively. Guidance to projects should be consistent with any enterprise-level standards, policies, or conventions, such as computing infrastructure, tools, or development practices.

A program may establish more specific guidance appropriate to its projects. This guidance may include prescribing a uniform development methodology (with section 2.0 content as a tentative basis), including process, quality criteria, and associated methods, practices, tools, and shared assets, that projects are expected to apply. A program may also direct appropriate collaborations across projects for coordinated provision and use of shared resources (such as people having specialized expertise, specialized equipment, or reusable software assets). All guidance should be expected to evolve as circumstances and technology change over time.

Data and Reality

The capabilities of a product are limited by the information accessible to it. Needed information is encoded in data obtained via edge and interface devices or by computations based on other data (e.g., to compensate for data that is not directly accessible, to maintain consistency of measured data, or to operate within alternative realities). The quality of a product's behavior is highly dependent on maintaining the quality of its data as a consistent and sufficiently precise, accurate, and timely representation of reality.

Developing a product encompasses deciding what information a product needs to accomplish its intended purpose, how to represent that information as data, and how to obtain that data as associated information changes. The product must then be built with the means to acquire, process, and make data available for computations that produce appropriate behavior, which in turn may initiate changes in reality. As customer needs evolve, changes in accessible information and capabilities for obtaining and processing data are significant factors in how a product is modified.

A key challenge for an evolving product is the character of the data that it needs to reference. As a product's capabilities evolve, the data it uses will evolve both in terms of what information is needed as well as how that information is to be represented. A concern in the proper design of software is having the means to modify how data is defined, represented, and converted. Deployment of a revised product can entail a significant effort to transform existing data to conform to the revised product's needs.

Information as represented in data can differ from reality due to uncertainty associated with limitations in how corresponding data is determined. Both devices and computations will be limited in the precision of data values produced and rely on assumptions concerning the accuracy of information sources. In addition, devices can have transient or fundamental flaws in its mechanism for measuring phenomena that limits the accuracy of its obtained data. Similarly, nominally correct computations can improperly modify precision or accuracy of data versus its actual value. Inherent latency in obtaining changing values can delay applying data updates. As a result of these issues, data is only an approximation of information defined by reality. This may warrant computational adjustments to avoid incorrect behavior resulting from physical limits of precision and accuracy in devices, latency in data acquisition, and precision and confidence levels in computations.

The means for operating with data is a platform capability for which software is able to define how data is organized/grouped and interconnected in such storage. (Data is discrete or aggregate values, structured into objects, and relationships between objects,

and may have associated semantics that specify how values and relationships are determined.)

For data integrity, a device may have associated metadata: precision or an error profile for the measurements it produces. A user may have an associated “credibility”. A derivation will have qualities derivative of the qualities associated with related data.

distinguish metadata from regular data based on usage, even if some data is also used as metadata (the organization of a document or its library ref nr is data, not metadata; metadata is (generally?) not accessible as product content): data that substantiates/justifies confidence in validity (integrity) of associated data. It provides a basis for explaining/justifying data plus evaluating the validity of referenced data and mitigating substantiated inaccurate, obsolete, or erroneous values, dependence on values that trigger recompute upon change, constraints on applicability of data. Metadata can be associated with data a system has about itself and about its containing environment. A system obtains relevant data about its environment directly via associated devices, indirectly via communications with related systems, or by derivation (computed or inferred) based on related data. Related data items may be aggregated into composite structures.

(techniques for cleaning up data values to be more accurate: data quality/integrity; methods of improving accuracy and precision over raw data; correct computation depends on valid data corresponding to reality: actual information) Data values can differ from associated information due to disparities in precision and accuracy of data collection or derivation mechanisms or latency between measurement and use of a resulting value. Data values may have an associated level of confidence in terms of how well it matches the value and currency of associated information, apply multi-source (multi-sensor, multi-perspective) data fusion to improvement accuracy of a value, or use interpolation to account for continuously changing values or future value projection and smoothing from past values.

Hardware Engineering

Basic software product engineering is provided with specifications for an operational environment, including the computational platform on which the software product is built to operate. Hardware engineering efforts may be undertaken prior to, concurrent with, or consequent to the software product engineering effort to build or acquire the hardware elements needed for the software to operate. Specifications of a device's properties and expected behavior may suffice for building a software-based emulator of the device with which a software product can operate in the absence or failure of the device's physical realization.

For a product that requires only off-the-shelf hardware as its platform, the hardware platform may be prescribed by the customer or specified as part of the software product engineering effort. In these cases, the platform hardware may be provided by the customer or supplied as an adjunct to software product delivery.

Device engineering

Physical reality is logically continuous and analog; edge and interface devices provide a portal between a discrete, digital computational platform and physical reality. Observations obtained via devices are inherently limited by the precision and accuracy of those devices. Devices provide measures of the environment that are digital approximations of the actual physical phenomena. Hardware devices are an information conduit between the analog physical environment and digital processing, to detect the state of the environment and to produce analog effects that transform that state.

Edge devices serve software as proxies for the operational environment, being the information space within which software operates. Interface devices provide the means for interactions/communications with users and external devices in the ecosystem. Other systems can be treated as the equivalent of a set of edge and interface devices through which information is shared and operations coordinated.

Devices (including any enabling software) are typically acquired from providers having expertise in the design, engineering, and fabrication or assembly of such devices. A

"virtual" device may be created by interconnecting a set of physical devices, with installed base software so as to provide other software with the illusion that the set is a single physical device.

Device engineering is generally a concern in conventional software engineering only in the narrow sense of choosing among available alternative devices. Often, the concern is only to build the software within constraints of what is in fact preemptively prescribed computational hardware. For more complex systems, systems engineering must identify capabilities that require a hardware realization, requiring either selection from among available devices or development of special purpose devices suited to specific needs. In other cases, it may be a matter of deciding whether there are specific capabilities that are feasibly implemented in software but would be beneficial to realize in specialized hardware.

A significant other aspect of device engineering is that modern devices are generally software-enabled analog-digital hybrids. Analog (sensor-effector) elements enable interactions with the physical environment; hardware realizations of functionality increase performance and predictability over software realizations; and software facets provide greater flexibility over hardware-only realizations. Encapsulating hardware device interfaces in software, is an established practice of software product engineering to avoid software dependencies on hardware realization details that may change.

Systems Engineering

A system emerges organically toward some perceived purpose based on the motivations and actions of entities operating interdependently in a shared environment. A system will be of interest to an enterprise if the purpose of the system is related to the enterprise's ability to achieve its objectives. An enterprise can attempt to induce beneficial changes in the behavior of a system by means of a product that will modify the behaviors of affiliated entities. Systems engineering is the effort needed to direct and coordinate expertise of relevant engineering and manufacturing disciplines so as to realize and employ such a product, balancing the concerns of each discipline.

Systems engineering employs software engineering at two levels: as the means for defining how the behaviors and interactions among affiliated entities are to be controlled and coordinated and as the means for enhancing the inherent physical capabilities of individual affiliated entities.

A systems engineering effort identifies the various elements of a system, how each element behaves, and how elements affect each other and their shared environment. This may entail actions such as building and deploying new devices, modifying existing devices, or modifying the practices that people follow as elements of that system.

More specifically, the role of systems engineering is to

- understand how the functionality for an envisioned improved system can be achieved with a feasible assembly of analog and digital elements and evaluate the most viable combination to develop;
- define the nature of system elements and boundaries between those elements and with interacting systems.
- allocate responsibilities for development of envisioned system elements based on competence in needed engineering disciplines;

Systems engineering will specify needed devices, facilities and resources for manufacturing those devices, and a physical infrastructure (i.e., a platform) in which those devices will operate. These will have associated quality criteria that in aggregate will enable the system to satisfy its objectives. The overall ensemble arises through a process of analyzing alternatives and tradeoffs to achieve a viable approach to achieving the envisioned system.

For software-based products built to operate using only conventional computing hardware, most aspects of systems engineering can be addressed in software engineering activities. However, lacking a proper systems engineering effort, some aspects, such as analysis of alternatives and quality tradeoffs, require increased consideration.

For more complex software-based systems (e.g., software operating in a dispersed cyberphysical system, a manufacturing or chemical process, or a hybrid physical/augmented-reality environment), a proper systems engineering effort provides needed context for building an appropriate software product. This effort, to be effective, must operate collaboratively with associated software engineering efforts, to ensure that implications of software alternatives and tradeoffs are properly considered.

Although the internal construction of a device can be opaque, it will typically take the form of software-enhanced hardware mechanisms. Software enables more complex functionality, better quality, and more flexible control of a device's operation. Furthermore, a device may be built to support the needs of different systems. As such, systems engineering can encapsulate the capabilities of each device within a software element that presents a tailored or enhanced view of the device's capabilities. As engineered systems become more complex aggregates of hybrid analog-digital devices and increasingly complex software elements, a comprehensive systems engineering perspective frames an effective software engineering effort.

Systems engineering will specify a platform into which software-based products can be injected to operate. A given software-based product can comprise both hardware and software components. A systems engineering-defined platform can integrate devices that are developed either independently or as integral agents of a software-based product. Either type of device can provide

The purpose of systems engineering is to make the operation of a system efficient and effective, both initially and over time as needs and circumstances change. The efforts of systems engineering that support this include:

- Conception – a formulation of the existing system and ecosystem, an envisioned improved system, and the implied effort required to transition toward the future system (current versus envisioned ideal; implied transition effort)
- Analysis of Alternatives and Tradeoffs – analyses of useful-life tradeoffs between alternative system realizations, considering relevant quality factors and

availability of required resources including technology, engineering and manufacturing competencies, materials, and finances

- Realization – allocation of tasks and resources to engineering and manufacturing efforts for the realization of needed system elements, associated manufacturing facilities and resources, and operational guidance
- Provisioning – the acquisition and supply of raw materials and processed parts needed for the realization and ongoing operation of the system and its elements
- Deployment – The means by which elements of the envisioned system, including software, are put into operational use and improved over time as needs and circumstances change
- Sustainment – the means by which elements are routinely monitored, maintained, and adjusted and by which worn, damaged, failing or otherwise deficient elements are repaired or replaced, including deployment of improved software, for the continuing effectiveness and efficiency of the in-use system
- Disposal – analysis of the efforts that will be needed to properly dispose of or recycle materials as the system evolves or reaches the end of its useful life, avoiding or minimizing any detrimental residual effects on the natural environment

Our attention here is directed to the facets of systems engineering that relate to realization and sustainment of a software-based product. Because the scope of a particular effort may target elements that make up only a portion of the system of interest, the product must account for constraints imposed by other aspects of the system. In most instances the envisioned system is a transformation of an existing system realized through the injection of the elements comprising the deployable product. Alternatively, the system as a whole may be a composition of multiple independently developed products, each possibly operating as an element of a distinct system in its own right. Understanding the behavior of the system as a whole requires understanding the separately realized but interdependent behaviors of these

"subsystems". This can be even more complex if the behavior of a system is built to be dynamically adaptive to changes in its operational environment.

Traditional linear approaches to systems engineering are unsuitable for development of modern continually-evolving systems. As an envelope for software engineering and software-based device engineering, systems engineering requires an iteratively collaborative approach that actively exposes uncertainty and anticipates implications of potential future change. This contrasts with traditional systems engineering approaches that treat software as a distinct concern rather than as the pervasive determinant of both system and device behavior that it actually is.

A challenge for systems engineering, given the increased importance of software in systems, is to recognize both that systems engineering decisions impose constraints on software engineering options and that software engineering decisions can impact systems engineering expectations about how the resulting system will behave. Systems and software engineering need to work together in a collaborative effort to ensure that decisions made are best for both. This entails both avoiding the tendency to impose changes on software engineering near the end of planned product development in order to compensate for unexpected hardware deficiencies and developing software that can be flexibly revised to adjust its behavior to suit actual hardware behaviors.

Beyond a narrow focus on achieving a single best solution given current needs and technology, systems engineering needs to explore and communicate concerning recognized uncertainties and potential change in needs and technology. An awareness of uncertainties and potential change allows software engineering to organize the software so that potential future changes will be easier and less costly. Software built without such information can be unpredictably difficult to change, leading to degraded structural integrity of its design unless additional effort to maintain that integrity is made as unforeseen changes are needed.

Product Implications for Customers

From a customer perspective, a product is a means to an end, enabling or improving the ability of the enterprise to achieve some objective. Customer use of a product entails

acquiring a product that provides needed capabilities, adjusting business operations to make best use of the product, and adapting operational practices and procedures as the product evolves, in accordance with changes in the customer's needs or otherwise imposed on the product by external circumstances.

Product Acquisition

(product acq framed as transition from curr to future operation)

Product acquisition is the procurement (purchase or licensing) of a product for deployment into and use by an enterprise. In some cases, product acquisition is a straightforward selection and procurement of an existing product based on providers' marketing efforts. When acquiring a product to address a well-defined customer need, the acquisition may entail only an effort to evaluate competing products against acceptance criteria and transition the customer enterprise to institute use of the selected product.

More generally, acquisition is a coordinated effort of an enterprise to identify and deploy products that will benefit the operation of the enterprise. An enterprise, based on a determined need for a product, may either institute a formal acquisition authority or designate management of an existing program or project as its representative to work with potential product providers. Prior to establishment of a provider project targeting a customer, provider program marketing works with each potential customer to establish the fit between the customer's needs and the provider's product scope and competence.

A designated acquirer serves as a conduit for all communications between the customer enterprise and product providers. The acquirer, ideally with the assistance of providers, will:

- determine the purpose and capabilities for a needed product,
- determine product acceptance criteria (to be modified as needed to account for capability, cost, schedule, or quality tradeoffs among candidate products),

- identify existing products or engage with potential providers to build prototype candidates, as appropriate, to evaluate their fit to need,
- Evaluate candidate products for comparative adherence to acceptance criteria with tradeoffs,
- Manage procurement of selected best-fit product, pending final evaluation encompassing any negotiated improvements,
- Orchestrate deployment of selected product into operational use, including adjustment of operational facilities and procedures, configuration and installation of the product, and provision of user documentation, training, and assistance.

Organizational Transition

For effective and efficient use of a product, the customer's operational processes and practices need to account for the role that the product is meant to play in the enterprise. In instituting the use of a new or improved product, the customer must consider whether and how operations should be revised.

Over the lifecycle of a product, there can be a recurring need to revise operations to make best use of the capabilities of the product. Ideally, products are selected or able to be tailored for a proper fit to existing enterprise operations. However, a new or improved product may offer (or impose) an opportunity to beneficially modify aspects of how the enterprise operates.

Current enterprise operations may entail imposing constraints on product acceptance or making beneficial changes in enterprise operations. As part of product acquisition, the enterprise should invest effort in envisioning how its operations may be beneficially modified. Such revision requires plans to communicate with and train personnel, with product provider support as needed, in how their work practices are to change with use of the product.

Product Evolution

A product will evolve over its useful life, gaining improved capabilities, adjusting to technology changes, or responding to evolving customer or market needs. One consideration in the development of a product is *sustainability*, the means by which the product is built to accommodate its subsequent evolution over its intended useful life. In a similar vein, customers need the means to prepare for and accept changes to employed products, both as their needs change and to maintain conformance with advancing technology or market conventions. Sustainability is enhanced to the degree that customers can conjecture to providers about the nature of potential future changes in their needs and operational environment.

Accommodating Customer Circumstances

The essential determinant of the value of a product is the degree to which the targeted customer perceives the product to be providing capabilities that fit their needs. A provider must accommodate this perception from two perspectives: the competence-based ability the program has to build a product that fits customer needs, initially and with changes over time, and the ability the customer has to adjust their operations to use that product. The initial determination of such accommodation is a primary responsibility of the marketing function of program management in identifying potential customers. Based on that determination, achieving a shared understanding of actual customer needs and building a conformant product is a normal aspect of development.

The effort required for a customer to adopt a product must be commensurate with the perceived value of the capabilities it provides. A product must either be built to fit well with a customer's operational practices and constraints or offer sufficient benefit to justify the effort required to change those. An initially constrained version of a product that is progressively enhanced over time may ameliorate the difficulty of the customer having to change how they work.

Shared understanding of needs requires establishing a collaborative relationship between provider project and customer in which the shared perception of those needs are iteratively refined, influenced by any practical limitations on the product that can be

cost-efficiently and timely built. Any disparity between actual needs and the provider's ability to build a suitable product at reasonable cost can be resolved if those needs can be revised so that a suitable product can be built. A project's product delivery element addresses the technical aspects of these issues while the program management element addresses the business aspects of negotiating a mutually agreeable revision of customer product acceptance terms.