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 a product to customers, and the perspective its customers have about a 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.

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 product development programs are 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
- Accommodation of customer circumstances over a product lifecycle (acquisition as a customer-side complement to product development, organizational transition, and customer implications of a changing product over its useful life)

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 executive level of management is responsible for enterprise governance, defining mission-directed objectives and strategy, obtaining additional resources (funding, people, facilities), imposing uniform practices and criteria, instituting administrative services, and monitoring and reporting on its activities.

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. Each program is chartered 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 designated to have responsibility for addressing the needs of a specified customer/simple market. A project chartered as a product developer would be responsible for managing an ongoing relationship with its designated (internal or external) customer to create and evolve a product that meets their needs.

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, influence the ways in which market and customer operations are likely to change. To the degree that customer needs are similar, program management may seek to leverage work across its 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.)

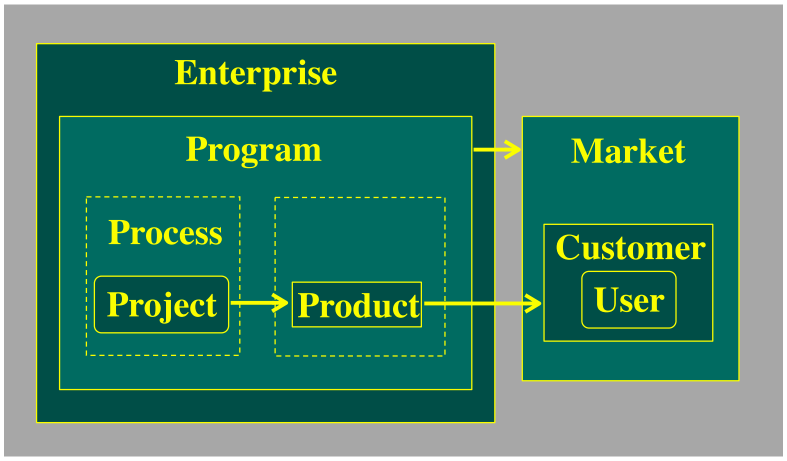

Figure 2.1-1. Program-Market and Project-Customer Relationships

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 possible 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

Data, being only an approximation of information, must be nominally sufficient to suit a product's purpose. Developing a product encompasses deciding what information is needed to accomplish its intended purpose, how to represent that information as data, and how to obtain that data as associated information changes. Data can represent not only observable information about the behavior of the containing environment and associated entities but also internal information about the operation and health of the product itself and its platform (e.g., for diagnostic detection and mitigation of defects or retrospective auditing of its operation).

A product's behavioral quality is dependent on maintaining the quality of its data as a proper representation of reality (based on accurate and timely information). Developers must make, and verify, assumptions regarding whether accessible data are sufficient to support correct product behavior. A product must 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 corresponding information.

Data, with supporting metadata, can be derived through computations that serve any of various purposes, such as to (1) translate between multiple representations of data (e.g., analog versus digital or numeric versus symbolic), (2) correct for imprecision, delays, or flaws in entity-provided or derived data, (3) maintain consistency among logically related data, (4) combine values of varying quality obtained from multiple sources (data fusion), (5) virtualize the behavior of an unavailable or inaccessible device, (6) smooth or interpolate data based on past values or theoretical criteria, (7) determine confidence in the validity of its value as derived, or (8) project the effects of conjectured changes in (actual or virtual) ecosystem information.

Metadata representing the provenance and quality of referenced data can be used to determine or improve the accuracy (e.g., fusion of data from multiple sources) and timeliness (e.g., interpolation or smoothing of a series of values) of obtained data, using computational methods appropriate to data attributes. Metadata may guide selection among computational methods appropriate to the data. It can also support explaining and justifying how values have been determined. Metadata may associate the degree of confidence (or credibility) in data values depending on their sources or determination.

As needs that a product addresses change over time so also does the information it needs and how that information is represented as data. When such changes occur, either the representation of current and retained data needs to be modified or the product must be modified to operate with multiple representations of the same information.

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 virtualization of the device with which a software product can operate pending availability or in the absence or failure of the device's physical realization.

Hardware engineering is typically 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, a systems engineering effort will 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 some cases, there may be specific capabilities that are feasible to implement in software but would be beneficial to realize in specialized hardware.

For a product that requires only off-the-shelf hardware as its platform, the hardware 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. When specialized devices are needed to satisfy customer needs, product development can entail both software and device engineering efforts.

Device engineering

Edge and interface devices are proxies by which software interacts with its environment and associated entities. Edge devices monitor physical phenomena to obtain measures that are approximations of physical reality and may induce physical effects on that environment. These devices are information conduits between a discrete, digital computational platform and the continuous, analog physical reality. Observations obtained via devices are inherently limited by the precision and accuracy of those devices. Similarly, interface devices provide the means by which software can share information and coordinate activity with entities operating independently in its environment.

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 coordinating a set of physical devices, with associated software, to provide client software with the perception that the set is a single physical device. Analogously, a single multi-function device may be exposed to software as multiple logical devices.

Modern devices are generally software-enabled analog-digital hybrids. 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, software engineering can encapsulate the capabilities of each device within a software element that presents a tailored or enhanced view of the device's capabilities.

The analog (sensor-effector) elements of a device enable interactions with the physical environment. In addition, hardware realizations of functionality can increase performance and predictability over otherwise equivalent software realizations. However, software provides greater flexibility over hardware-only realizations for enabling timely changes in device behavior or adaptivity to operational circumstances. Analogously, 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. Such encapsulation also ameliorates the tendency to rely on unplanned changes in software to compensate for divergences or deficiencies in actual as-built versus as-specified hardware behavior, localizing any necessary changes in the encapsulating software and shielding referencing software from unforeseen changes.

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 products, balancing the concerns of each discipline.

A given software-based product can comprise both hardware and software components. A systems engineering-specified platform can integrate devices that are developed either independently or as integral agents of a software-based product. Systems engineering specifies the platform on which a software-based product will operate.

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 products (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.

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 entities within that system. 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.

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 and expected behaviors of subject 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.

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

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, an envisioned system is a transformation of an existing system, realized through the injection of the elements comprising the realized 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.

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. In the development of modern constantly evolving systems, software is a pervasive determinant of both system and device behaviors.

Systems and software engineering need to iteratively collaborate to ensure that decisions balance both perspectives, focusing on uncertainties, tradeoffs, and implications of potential future changes in needs and technology as these related to all relevant engineering disciplines. An awareness of uncertainty and potential change allows software engineering to build products for which considered changes will be easier and less costly. Software built without such information can be unpredictably difficult to change, and worsening over time, unless additional effort to maintain structural integrity is made as unforeseen changes are encountered.

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. 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 the product that can be viably built. 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 revised over time may ameliorate the difficulty of the customer having to change how they work.

A proper understanding of actual needs benefits from a collaborative relationship between provider project and customer agent in which a shared perception of needs is 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 capabilities of a product that a provider can build with reasonable effort may require reducing needs to match feasible product capabilities.

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 is changed over time.

Product Acquisition

Product acquisition is the procurement of a product for deployment into and use by an enterprise. In some cases, product acquisition is the 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 (including any customization options) against acceptance criteria and transition the customer enterprise to institute use of the selected product.

More generally, product acquisition is an element of a transition in the operations of an enterprise, intended to achieve specified improvements. In this, an acquisition is a coordinated effort to identify and deploy products that will support that transition. An enterprise, based on a perceived 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 product providers.

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

- specify the purpose and general capabilities for a needed product,
- verify with marketing of each potential provider that its product scope and competence is a suitable fit for the customer's perceived needs
- 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 an acceptable fit to the need,
- Compare candidate products for adherence to acceptance criteria with tradeoffs,
- Manage procurement of a selected product through delivery and evaluation of the product for acceptance to operational use,
- Orchestrate deployment of an accepted product, 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. Enterprise or externally-imposed conventions may constrain how operations can be changed and whether a product can be accepted. If operations can be changed, a rethinking of how to operate due to a product deployment is needed as part of the transition and requires plans to communicate with and train personnel in how work practices are to change.

Product Evolution

A product will evolve over its useful life, whether 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 legal and market conventions. Sustainability is enhanced to the degree that customers can speculate with providers as to the nature of any potential future changes in their needs and operational context.