1.4 A Vision for the Engineering and Manufacture of Software

Based on the precepts, terminology, and foundational concepts presented in preceding sections, this text defines a methodology for the systematic engineering and manufacture of software. This methodology is conceived as the means to realize a capabilities-based vision for the building and evolution of customized software-based products[1].

This text presents two distinct but conceptually related approaches to building software, one following basic software product engineering practices and the other extending that approach to a realization of product family engineering as the means for manufacture of similar customized products. Both approaches are seen as amenable to a particular type of automated support for the anticipated customization of products to specific needs. As software techniques and technology change over time, the form this methodology and associated practices take will evolve.

The Essential Vision: Producibility

A product is conceived to provide capabilities that enhance a customer's ability to achieve their objectives. *Producibility* is ***the ability to deliver needed capabilities in a timely, cost-effective and predictable manner***. This can be thought of here as having a proper ability to manufacture software-based products.

This ability has five elements:

- (Analysis) The ability to determine and economically specify the behavior (capabilities and qualities) that an envisioned product should exhibit to serve its intended purpose
- (Synthesis) The ability to efficiently build a product as specified, resolving engineering tradeoffs as needed to best approximate specified behavior

[1] This vision extends from a 2007 research and transition initiative under the auspices of the Office of the Secretary of Defense: [*Software-intensive Systems Producibility: A Vision and Roadmap (v 0.1)* CMU Software Engineering Institute (CMU/SEI-2007-TN-017), Dec 2007] and summarized in G. H. Campbell, "Advancing Producibility for Software-intensive Systems", Software Tech News 11 (4), Dec 2008, 13-17.

- (Evaluation) The ability to evaluate the value (utility and quality) of a product as specified and as built
- (Transition) The ability to deliver a product and adjust the operation of a customer enterprise to accommodate its effective use
- (Evolution) The ability to anticipate and efficiently revise a product's behavior as needs, circumstances, and technology change

The ability to manufacture a product is itself based on being able to envision and realize a suitably efficient and effective means of production. Although such means can begin as systematic tool-supported human effort, the vision is to achieve an automated medium for human-guided production. As a software-based product requires both hardware and software elements as well as associated operational practices, all must be realized in consistent form for the realization of a complete product suitable for use in a specified context.

The manufacture of hardware, both platforms and devices, has already advanced somewhat in the direction of the producibility vision, in the forms of mass customization, computer-aided design and manufacturing (CAD/CAM), and 3d printing. The focus of this text is on producibility that extends these hardware advances with an integrated hardware-software-system formulation for building complete software-based products. This particularly entails moving progressively toward an ability to specify and build a software-first realization of system-level capabilities, from which elements can be realized in general- and special-purpose hardware as needed.

The methodology presented in this text offers an evolved conception of a comprehensive approach to achieving the producibility vision. This vision encompasses three levels of capability, starting with a systematic approach to building singular evolving software products, extending to an approach for building multiple similar but customized software-based products, and advancing over time by exploiting emerging technologies to achieve an optimally streamlined capability for the engineering and manufacture of customized products.

The Engineering–Manufacturing Combine

In the producibility vision, manufacturing is the effort entailed in building a product for a simple market. The objective of a manufacturing effort is to build a product that provides capabilities needed by its customer(s). Engineering is the effort to create a capability for building products that manufacturing efforts can use for enhanced productivity and product quality. The objective of an engineering effort is to provide associated manufacturing efforts with the means to efficiently and effectively build products that fit the needs of a designated coherent market.

The initial product realization effort is a collaboration between an engineering effort and one or more manufacturing efforts. Initial manufacturing efforts may undertake some foundational tasks of the engineering effort, based on competence gained in the engineering of legacy products. As the engineering effort proceeds, it solidifies the problem-solution space that represents the long-term needs of the program's overall targeted market and creates a corresponding streamlined capability that each manufacturing effort will thereafter use to rapidly build and evolve a customized product for its customer. As addressable market composition and its needs evolve, the engineering effort will evolve the manufacturing capability to support building new and revised products.

Because an engineering effort is conceived in support of multiple potential manufacturing efforts, it is the means by which those efforts are guided to conform with common practices. It enables optimum commonality in the way products are built and the capabilities and quality of those products, recognizing the diversity in customer needs that must be considered. In turn, the engineering effort works to collaboratively identify and prioritize the future needs of those manufacturing efforts to address the changing needs of their customers (i.e., the market as a whole). This may entail limits on some products that would conflict with the objectives and scope of the overall program.

Producibility Realized

Producibility has three aspects: developer productivity, product value, and customer acuity. Developer productivity is a function of process quality and developer

competence. Product value is a function of product utility and product quality. Customer acuity is a function of insight concerning current and foresight concerning potential future needed capabilities. The following identifies how each of these may be addressed in a realization of the producibility vision.

Developer Productivity

- All information about a problem and its solution is expressed in a comprehensive model of the product. Questions about the product realization or behavior can be answered by interrogation of the product's model without direct study or operation of the product itself.
- An approximation of the envisioned product, with a subset of needed capabilities, is to be built within 1-3 months of manufacturing project initiation. This and subsequent revisions will provide a concrete basis for refining understanding of customer needs and identifying focus on product quality elements.
- The behavior of a product is described in terms of its interactions with a specified (physical, virtual, or hybrid) ecosystem. The behavior of the environment and each associated entity may be simulated to provide an operational context for evaluating the product as it is being developed. Any hardware components of the product will be software-encapsulated (including possibly emulated hardware elements). The product may be built with instrumentation for monitoring and control of the operation of components to support analysis and verification of the product's behavior under prescribed normal and anomalous conditions.
- Software, hardware, and systems engineering disciplines are unified in a collaborative engineering approach, using appropriate methods, tools, and practices, to enable the systematic automated manufacture of complete customized software-based products.

- Product engineering efforts standardize well-understood aspects of problems and solutions for a targeted market so that product manufacturing efforts can focus on less well understood or unsettled aspects of each customer's needs.
- Product engineering iteratively evolves manufacturing capabilities such that manufacturing efforts have a stable baseline capability that is improved and extended on a predictable schedule.
- Product manufacturing iteratively builds a product such that its capabilities can be demonstrated on demand in a artificial ecosystem, as proof of concept and subsequent progress to the customer, changing as work progresses and understanding and expression of customer needs improves.
- Effort required to manufacture a product is predictable based on tradeoffs among project productivity, product quality, cost-schedule, and externally imposed constraints, adjusted for product model complexity.
- Product engineering provides the means to modify an existing product model in order for product manufacturing to efficiently build a revised product when customer needs or operational circumstances change.
- Product manufacturing offers customers the option of an incrementally-built product, providing a series of predictably timely, capability-subset releases, for progressive commitment, cost control, and managed evolution of enterprise operations.

Product Value

- All products reflect standardized product engineering practices and shared levels of subject-matter and technical competence.
- Every product is built to address the specific needs of each customer, contingent on fit with program-specified market coherence and technical competence.
- All products satisfy a specified level of quality criteria based on the nature of customers' needs and the program's level of engineering competence. Relevant measures of product quality are estimated during manufacture using

engineering-provided analytic and empirical techniques to identify weaknesses and tradeoffs in considering problem-solution alternatives.

- Determinations of product value (both utility and quality) entail a mix of evaluation techniques, including peer and expert reviews, analytic techniques, and empirical techniques (e.g., testing of the product's behavior against expected results).
- Product quality is improved when defects are discovered and resolved but the quality of a product is evidenced only in part by the lack of unresolved defects. Any discovered defect is used as the basis for a root cause analysis to determine the source of the defect and modify practices to ensure either avoidance or early detection and correction of any similar future defects.
- Products are cost-effectively evolved as customers' needs change over time.
- All affected products are collectively revised to address any improvements in capability or discovered defects.

Customer Acuity

- Customers identify products to acquire based on provider descriptions of the general capabilities of a prototypical product they offer and how instances can be customized. Providers help customers determine product fit to their needs and identify any product customizations or enterprise operational changes needed.
- Customers articulate and refine their needs through a provider-assisted standardized decision process of exploration and resolution appropriate to the problem-solution space of a provider's ability to build products.
- Alternative versions of a product can be specified and generated to evaluate different resolutions of problem-solution uncertainties.
- Potential changes in enterprise practices can be explored in alternative versions of user roles and practices that an envisioned product may enable.

- Customers identify potential changes in their operations that would motivate changes in needed product capabilities, allowing developers to evaluate and perform corresponding engineering efforts in anticipation of such changes.

Supporting Perspectives

- A program charters development projects with consideration of business objectives, market focus, and demonstrated competence in building products appropriate to its targeted market. This implies limitations on the products that can be effectively and efficiently built; changes in these factors imply changes in what products can be built.
- Products are developed to support the needs of customers in a program-targeted market, or possibly other endeavors of the program itself. In progressing toward release, products are iteratively improved and incrementally delivered, resulting in a series of subsets and variations. Changes in program focus may lead to changes in developer practices.
- The practice of incremental development is a realization of the concept of product subsetting. A product that is a partial solution to a particular problem may be a complete solution to a simpler problem. Analogously, a product that is a complete solution to a particular problem may suffice as a partial solution to a more complex problem. An implication of this is that an explicit tradeoff can be offered between product capabilities and time to build: a customer can be offered an interim incomplete solution to their needs, satisfying a coherent subset of those needs, with the expectation of a more complete solution in due course.
- A product is rapidly revised as understanding, customer needs, technology, or circumstances change. A modified product is derived as a result of modifying an existing product model. There is no substantive difference between creating a new product and modifying an existing product for a new or existing customer. In all cases, the result is a suitably customized product that addresses the particular customer's needs.
- A provider program and its targeted market must coevolve. The program works to build products that will meet current and future customer needs. The market

in turn will organize its operations to take advantage of the changing capabilities of available products.

- A development environment is a software-based product that provides capabilities for building software-based products. Such an environment is characterized in terms of the type of products that it can be used to build. These products are characterized in terms of perceived similarities in customer problems and their potential solutions, in terms of common and differing needs. By limiting an environment to building only similar products, the effort needed to build a product can be reduced by eliminating redundant effort required to repeatedly develop portions of products that do not differ and limiting the alternatives for portions that do differ. If capabilities needed by the market change, the environment is able to be modified to build products having those capabilities.
- A product is an approximate solution to an approximate problem (i.e., one that is inadequately defined due to missing or uncertain information). Customer needs is an under-constrained specification of the capabilities that a product should support (i.e., many different products may conform to them); the job of developers is to arrive at an appropriately over-constrained specification of the expected behavior of a best-fit product; validation is determination that requirements defines a product that satisfies customer needs.
- Product development pursues a balance among behavioral functionality and quality, contextual and solution constraints, and cost-schedule predictability. The value of a product is its utility and quality in providing the customer with cost-effective and responsive capabilities. Products are evaluated relative to the customers' criteria for value. Different tradeoffs in product value can result in differences in functionality and quality factors.
- An enterprise to achieve its objectives depends on the efficiency and effectiveness of its operations. This can entail changes in how the enterprise operates, enabled by the use of appropriate products that provide capabilities suited to that operation. Needed product capabilities are not always easily communicated,

being initially incomplete or poorly understood and likely to change, both during and after initial development. Furthermore, insights gained both in building a product and in the subsequent use of a deployed product can lead to a better understanding of actual needs. Determining actual needs may require identifying, exploring, and resolving uncertainties and tradeoffs.

- Customers perceive the need for a product in terms of the capabilities it gives them toward meeting their objectives. However, a customer may have gaps, uncertainties, or conflicts regarding their actual needs and may not be able to clearly communicate those needs even as they do understand them. As a result, they may not know exactly what capabilities a product should provide. Developers help customers articulate needs and resolve gaps, uncertainties, and conflicts, see how choices will affect functionality, quality, and cost of a solution, and systematically evaluate tradeoffs among potential solutions. Resolving uncertainties and tradeoffs includes building partial alternative solutions that customers can evaluate and compare.
- Developers have competence in the relevant problem-solution space, as a result of previously building similar products; this enables them to expedite building the right product with increased certainty. They are better able to communicate with customer representatives in order to understand and build products that support their needs. Developers without such competence have to spend additional time gaining needed knowledge and expertise.
- Developers require an understanding of how different customers perceive and express the same needs and how their needs may actually differ. Developers determine which differences a product must support, which ones customers can adjust their operations to fit, and which ones cannot be supported.
- A product is built with the expectation that its capabilities will be improved to meet changing needs over its useful life. Customers and developers must project likely future changes and how these will affect the product's development. A product is built in a way that makes potential changes less difficult, reducing total cost over its useful life.

- A software-based product is a holistic ensemble of hardware, software, and practices, enabling interactions with associated and external entities within a natural, artificial, or hybrid environment, to provide capabilities supporting operations of an enterprise. Hardware is principally an enabler, expediter, and conveyance of software capabilities needed by an enterprise. Software produces observable behavior in an ecosystem through the sensing and effecting capabilities of associated hardware devices. Hardware is also a means to more efficiently convey software-defined capabilities.
- Project documentation (whether hardcopy or electronic) of a product and its development serves two purposes: to define for current developers a shared view of how the product is being built and to communicate to future developers how the product was built as a guide to its rationale and an aid to making changes.
- When capabilities needed by multiple customers are similar, it is reasonable to construe them as forming a simple market so as to provide the same product to all. As long as their (changing) needs stay aligned, a responsively modified product may remain suitable for all. However, the needs of different customers can converge and diverge over time. When their needs differ, the differences can be essential or incidental. Incidental differences are those that a customer is able to resolve by modifying their practices to fit the product. Conversely, to address essential differences, unless the product includes a sufficient means to be configured on installation or use, the customer may need to obtain an alternative version that better fits their essential needs and circumstances.
- In addressing a coherent market, different products as well as the different versions of a product will be similar, differing in some ways but having much in common that can be commonly developed and evolved without duplicate effort. This reduces the effort to be spent on well-understood aspects of each product, leaving more time to focus on poorly understood, varied, or more complex aspects of each customer's problem and solution.

- Some aspects of "producibility realized" are more traditionally viewed as conceptions for the engineering and manufacturing of physical products. These apply for two reasons: (1) this approach envisions software-based products as subsuming traditional hardware and system concerns and (2) this approach envisions software as being realized through more disciplined engineering practices, ultimately taking a form that builds upon and extends modern manufacturing practices.
- Automated behavior is a hybrid effect of co-designed hardware and software capabilities. Computational platforms (hardware-software composites) are the means by which software produces behavior, using computational devices selected or designed to support particular software capabilities, including specialized processors that can expedite specific software behaviors. The platform is the medium through which software interacts with external entities (users/operators and other systems). Hardware devices are the conduits through which software interacts with its environment. A device may be defined to be either an entity in a product's ecosystem or a component of the product itself.
- Software is the means by which the behavior of a product is defined. Hardware is the medium through which software senses and interacts with its ecosystem. Software can be used to characterize the behavior of a hardware devices as realized through its physical mechanisms. A device can be built or selected to exhibit software-specified behavior. Software is a means for specifying capabilities to be realized in a device, model its expected behavior, and emulate its functionality when its physical realization is not available.
- A device may be hardware-only but is typically a hardware-software hybrid. The behavior of a hardware device may be augmented with encapsulated software to enable customization or future changes in that behavior. Conversely, critical software capabilities may be implemented in hardware to ensure attainment of specific quality criteria (e.g., enhanced responsiveness, accuracy, or security).
- A device may embed integral software-enabled capabilities that allow it to be functionally enhanced and flexibly adjusted, monitored, and modified or

transparently replaced over time as needs and operational circumstances warrant. In some cases, software can be built to operate as a surrogate for obtaining services or data from unavailable hardware devices.

- Software is the means by which all aspects of a product are first realized. The essence of this is virtualization through the software encapsulation of all behavior that a product will exhibit. This means that all aspects and elements of an envisioned product are first realized and explored in a "pure" idealized expression of behavior in software. The decision to transform any mechanism into a hardware realization is made either to enable interactions with the physical world or to make some aspect of product quality more predictable. Hardware components may have well-understood properties and interactions that support more accurate prediction of key product quality criteria.

Uniformly Representing All Entities as Logical Devices

{equivalence of all entities: people, devices, and aggregate entities (i.e., systems)}

All interactions of software with physical (or an equivalent virtual) reality, including the environment and all associated entities (i.e., devices, users, and systems or other aggregate entities), can be treated by software as encapsulated in a suitably defined logical device. From the perspective of a product, a logical device whether affiliated with or external to the product provides access to relevant data and actions associated with the information space of an ecosystem entity. Analogously, the information and mechanisms associated with an external system, enabled by a cohort of affiliated devices, can be represented and accessed by a software product as the equivalent of one or more logical devices. As such, all external access by a software product can be viewed abstractly as interacting with devices, with no special consideration given to whether the access is associated with an affiliated device, an external device, or a virtual device representing another system.

- Any product is able to be customized based on the specific needs, circumstances, and preferences of its customer, recognizing the need to appropriately address the sources of product change and market diversity. This encompasses the use of disciplined practices of a program-provided capability for systematically building either singular evolving products or customized instances of a market-associated product family.
- Decision-guided: Any buildable product corresponds to the resolution of alternatives represented by a designated set of characteristic decisions; these decisions represent differences among an envisioned set of addressable problems and associated potential solutions; each combination of decisions corresponds to a different product. (consequence: a product is the realization of a problem-solution pair that can be expressed in customer and developer choices.)
- expression of customer needs is simplified to require only a prescribed set of choices that resolve recognized uncertainties and tradeoffs that are representative of the targeted market for such products
- (incompleteness, uncertainty, and indecision) Ability to explore alternatives and tradeoffs:
- An ecosystem is an integrated information space: elimination of physical-virtual distinction / transparent treatment of all entities and environment as active/dynamic information spaces
- A product is created iteratively and incrementally based on what is known/well-founded, augmented by consideration of alternative possibilities at every level **<-->** an iterative incremental development approach supports discovery and correction of incomplete, inconsistent, and inferior aspects of a product model
- (resolving uncertainty; separation of concerns) Engineering is the building of models that enable decision making; associated rationale informs future changes when circumstances change. A model focuses on essential aspects of its subject, excluding incidental aspects, for objective comparison of alternatives. <-->

alternative expressions of a problem and its potential solutions can be built and both analytically and empirically evaluated.

- A product can be built to include mechanisms that enable observability of its internal behavior as a means for understanding the mechanisms underlying its observable behavior. both for analytic and empirical evaluation and for operational health monitoring, maintenance, and repair
- When needed, a product may be built to be modified (or self-modify/adapt to sw defects, hdw failures, or operational conditions) during, and without disrupting, its ongoing operation.
- a product is configurable based on resolution of key customer decisions to achieve a approximate best-fit given the products that the development environment is able to build <--> key determinant decisions are applied to resolve uncertainty, diversity, and change without requiring unforeseen reengineering effort (reeng at prod family vs prod level)
- Software is built to be independent of its actual physical realization. It may operate on its actual operational platform or a virtualized facsimile. It can work with any mix of actual or emulated hardware components of the product and actual or simulated behavior of users, devices, or related systems within an actual or simulated operational environment. <--> sw built to operate on indeterminate or variable computational technology; A product is built to operate on a computational platform that may take multiple forms or change
- Emulations and simulations are used in product verification and empirical evaluations of alternative solutions. These avoid the need to access external resources and also can be instrumented for observability of internal activity and experimentation with artificial instigation of failure conditions.
- Emulations and simulations may be part of an operational product's functionality (e.g., to suffice in place of a missing or defective device; to obtain data by alternate means; to mimic the behavior of inaccessible remote entities).

- Behavior of hardware or software components of a product may be tunable to support configurable or dynamic tradeoffs and predictability among quality criteria.