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 domain-specific realization of a product family for the derivation 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 additive fabrication (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

A producibility effort is a collaboration between an engineering effort and one or more manufacturing efforts. Manufacturing is the effort entailed in building a product that provides capabilities needed by its customer. Engineering is the effort to create a capability for building products that manufacturing efforts can use for enhanced productivity and product quality.

Competence gained in relevant past development efforts provides an initial basis, including candidate legacy assets, for the engineering effort. The engineering effort proceeds to actualize a bounded problem-solution space that embodies the long-term needs of the program's targeted market—building a streamlined capability that each manufacturing effort can use to rapidly build and evolve a customized product for its customer. As addressable market composition and its needs evolve, the engineering effort will refine, extend, and 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 as well as the capabilities and quality of those products, recognizing any diversity in customer needs that must be accommodated. 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 also entail limitations on the ability to build products having capabilities that are not compatible 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 capabilities needed and

foresight concerning potential future capabilities. The following looks at how realizations of the producibility vision will address each of these.

Developer Productivity

- Software, hardware, and systems engineering disciplines are unified to form a collaborative engineering approach, using appropriate methods, tools, and practices, that enables the systematic automated manufacture of complete customized software-based products. An engineering effort is a capital investment in the organizational competence needed to build products responsive to a targeted market and standardizes well-understood aspects of problems and alternative solutions for that market to create a manufacturing capability.
- An engineering-provided manufacturing capability enables manufacturing efforts to focus on resolving less well understood or unsettled aspects of each customer's needs so as to predictably build and evolve a responsive product. Engineering iteratively evolves the manufacturing capability such that manufacturing efforts have a stable baseline capability that is improved and extended on a predictable schedule in keeping with evolving market needs and technology.
- Manufacturing specifies a product in a product model that is iteratively revised as understanding, customer needs, technology, or circumstances change. This iterative refinement, in turn, provides a concrete basis for better understanding customer needs and identifying and resolving uncertainties, alternatives, and tradeoffs in product capabilities and quality.
- A manufacturing effort has the means to incrementally build and modify a product, with a series of predictably timely, capability-subset releases, for progressive commitment, cost control, and managed evolution of customer operations. An approximation of an envisioned product, with a subset of needed capabilities, can be built quickly (e.g., within 1-3 months) after manufacturing project initiation and subsequently revised thereafter over its useful life.

Product Value

- A product is built to address the specific needs of a customer or simple market. A products is evaluated relative to the customer's criteria for value: its utility and quality in providing cost-effective and responsive capabilities.
- The value of a product benefits from the provider having standardized development practices, tailored to the type of products to be built, and appropriate levels of subject-matter and technical competence. Product development continues over the useful life of a product, both to allow timely delivery of needed capabilities within (time-cost-competence) constraints and to accommodate resolving uncertainties and addressing changes in needs.
- Product value is a balance among interacting behavioral quality factors, extrinsic and intrinsic constraints, and developer cost-schedule goals. Relevant measures of product quality factors are estimated during manufacture using engineering-provided capabilities to identify tradeoffs among problem-solution alternatives.
- Achieving expected product value is a multi-faceted endeavor, relying on directed peer and expert reviews of individual element content and for consistency in content of related elements, analytic techniques for predicting quality factors across elements, and empirical techniques for measuring and verifying predicted value experimentally and in operational use. Effort needed to achieve product value is reduced when previously developed assets of known value can be used in building a product.
- 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. A discovered defect is the basis for an engineering 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.
- Both existing and future products benefit from engineering improvements in manufacturing capabilities. Existing products can be cost-effectively revised

using improved manufacturing capabilities to provide enhanced product value, with or without changed needs.

- Product value is understood in terms of its interactions within a specified ecosystem. Ecosystem composition and behavior may be simulated to provide a facsimile operational context in which the product, instrumented for monitoring and control, can be evaluated under prescribed normal and anomalous conditions as it is being developed. This can also be used to demonstrate incremental progress toward realization of needed capabilities.

Customer Acuity

- Customers perceive the need for a product and value it in terms of the capabilities it gives them toward meeting their objectives. The customer's perception of their needs may initially be incomplete, uncertain, or poorly understood, not easily communicated and likely to change both during and after initial development of a solution. A developer having complementary problem-solution competence (i.e., the ability to build corresponding products) focuses on resolving gaps, uncertainties, and conflicts in perceived needs to characterize a product that meets actual needs.
- Insights gained in experimental use of a candidate product can lead to better understood or changed perception of needs. Alternative versions of a product can be specified and built to explore tradeoffs with different resolutions of problem-solution uncertainties.
- Potential changes in enterprise practices can be explored in alternative versions of user roles, automation, and practices that an envisioned product could enable. New or revised product capabilities may expose opportunities or issues to beneficially modify those practices as well. Customers may identify potential changes in their operations that are enabled by changes in product capabilities, allowing developers to evaluate and perform corresponding engineering efforts before such changes are needed.

- Products are built with the expectation that they will be changed to provide improved capabilities as needs and circumstances change over time. When customers and developers can project potential future changes in capabilities and supporting technology, products will be built in a way that makes such changes less difficult, reducing total development costs over their useful lives.

Supporting Perspectives

Producibility is achieved through iterative repetition in the engineering, manufacture, and use of products. Some perspectives inform how and why producibility-motivated practices will lead to improved development of effective software-based products.

- A program bases its development efforts on business objectives, market focus, and demonstrated competence in building products. These imply limitations on the products that can be efficiently and effectively built but a program and its market can coevolve for mutual benefit. A program builds products that meet current market needs and will evolve to build products that meet future needs; the market, in turn, will evolve its operation to take advantage of improved product capabilities.
- A product is properly viewed as being an approximate solution to an approximate problem (i.e., one that is imprecisely defined due to incomplete or uncertain information) and therefore likely to change not only over time but during development. Customer needs represent a partial, under-constrained specification of a product (i.e., that many different products would satisfy); the developer produces an over-constrained specification of a buildable, needs-conformant product.
- Software is a means by which all aspects of a product can first be concretely realized. The means for this is the virtualization through software encapsulation of all behavior that a product will exhibit. 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 realize any behavior in

hardware form is made either to enable interactions with a physical ecosystem or to gain the improved predictability of hardware behavior.

- A coherent market is one in which different products, as well as the different versions of each 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.
- Developers having competence in the relevant problem-solution space, particularly as a result of previously building similar products, are able to more rapidly build the right product with increased certainty. They are better able to communicate with customers 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.
- In building multiple similar products for a coherent market, developers gain an understanding of differences in how customers perceive and express the same needs and how their needs may actually differ. With this understanding, developers can determine which differences products must support, which ones customers can adjust their operations to fit, and which ones cannot be reasonably supported.
- Project documentation (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, alternatives considered, and rationale for decisions taken as a basis for making future changes.
- The practice of incremental development is analogous to 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 a tradeoff can be offered between product capabilities and time to build: a customer may accept an interim solution that satisfies a coherent subset of their needs with the expectation of a more complete solution in due course.

- 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 types of products that it can be used to build. By limiting a development environment to building only similar products, the effort needed to build a product is reduced by eliminating redundant effort required to repeatedly develop portions of products that do not differ and by limiting the alternatives for portions that do differ. As capabilities needed change, the environment is modified to support building products having those capabilities.
- Product 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. The platform is the medium through which software interacts with external entities (devices, users/operators, and other systems). Entities are the conduits through which software interacts with its environment.
- Software is the means by which the behavior of a product is defined. From a software perspective, an ecosystem is a unified information space representing the integrated behavior of a set of entities within a shared environment. A software-based product interacts with an ecosystem by means of associated entities that inform, enable, expedite, and convey software-defined capabilities supporting the objectives of a customer enterprise.
- Hardware is the medium through which software interacts with an ecosystem. Software can be used to characterize the behavior that is to be realized through the physical mechanisms of a hardware device. A device can be built or selected to exhibit software-specified behavior. Software can be used to specify the capabilities of a device, model its expected behavior, and emulate its functionality when its physical realization is not available.

- Software is built to be independent of its computational platform. It may operate on a physically realized platform as specified or a virtualized facsimile. It can work with any mix of actual, emulated, or simulated entities within an actual or simulated operational environment.
- A hardware device can be software enabled. Encapsulated software can be used to customize, enhance, or enable changes in its behavior. Similarly, critical software capabilities can be implemented in hardware to ensure attainment of specific quality criteria (e.g., enhanced responsiveness, accuracy, or security). Embedded software can be used to functionally enhance and flexibly adjust, monitor, or modify or transparently replace a device over time as needs and operational circumstances warrant. In some cases, software can be built to operate as a surrogate for effecting services or data from unavailable hardware devices.