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 to build singular evolving products 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, this methodology and associated practices may 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

The benefits of producibility are realized in three dimensions: 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 enterprise competence needed to build products responsive to a targeted market. This investment focused on the needs of a coherent market provides leverage for rapidly building similar high-quality products.
- An engineering-provided manufacturing capability standardizes well-understood aspects of problems and associated alternative solutions that address the needs of the targeted market. This capability allows 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 provides manufacturing efforts with a stable baseline capability that is iteratively improved and extended on a predictable schedule in keeping with evolving market needs and technology.
- A manufacturing effort incrementally builds and modifies a product, with a series of predictably timely, capability-subset releases, for progressive commitment, cost control, and transition 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 of project initiation) and subsequently revised over its useful life.
- Manufacturing specifies a product in the form of a product model that is revised to reflect changing understanding, customer needs, technology, or other circumstances over its useful life. This iterative refinement, in turn, provides a concrete basis for improved understanding of customer needs and for identifying

and resolving uncertainties, alternatives, and tradeoffs in product capabilities and quality.

Product Value

- Each product is built to address the specific needs of a customer or simple market. A product is evaluated relative to the customer's criteria for value: its utility and quality in providing cost-effective and responsive capabilities.
- 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.
- Product value 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, to allow timely delivery of needed capabilities within (time-cost-competence) constraints as needs change.
- Product value is a tradeoff among needed capabilities, 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 comparatively evaluate problem-solution alternatives.
- Achieving expected product value is a multi-faceted endeavor, relying on directed peer and expert reviews of product model content for consistency among related elements, analytic techniques for evaluating quality factors, and empirical techniques for measuring and verifying expected value in experimental and operational use.
- 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.

Discovered defects are the basis for engineering root cause analyses to determine their source and modify practices to ensure either avoidance or early detection and correction of any similar future defects. Effort needed to achieve product value is reduced when engineering-provided assets of known value are used in building a product.

- 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.

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 focuses the effort 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 a changed perception of needs. Multiple versions of a needed product can be specified and built to explore tradeoffs with alternative 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 challenges to beneficially modifying 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 the customer commits to such changes.

- 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 can be built in a way that makes such changes less difficult, reducing total development costs over each product's useful life.

Supporting Perspectives

Producibility is achieved through iterative repetition in the engineering, manufacture, and use of products. Some additional 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.
- Developers having competence in the relevant problem-solution space, particularly as a result of having previously built 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 experience have to spend additional time gaining needed knowledge and expertise.
- 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 even 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.

- Product subsetting can be viewed as a complement to incremental development, removing versus adding capability. 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. A key implication of this is that the developer may be able to offer the customer a capabilities versus time-to-build tradeoff: accept a working partial solution to a coherent subset of their needs with the expectation of a more complete solution in due course. In any case, this is a basis for concretely demonstrating progress toward completion, verifying that an unfinished product satisfies some portion of customer needs.
- 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, including alternatives considered and rationale for decisions taken, as a basis for making future changes.
- A coherent market is one in which different products, as well as the different versions of each product, will be similar, differing in certain ways but having much in common that can be commonly developed and evolved without duplicate effort. Less time need be spent on well-understood aspects of each product, leaving more time as needed to focus on poorly understood, varied, or more complex aspects of each customer's problem and solution.
- 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.

- A development platform is a software-based product that provides capabilities for building software-based products. Such a platform is characterized in terms of the types of products that it can effectively be used to build. By focusing a development platform on the means to build only similar products, developer effort can be streamlined, eliminating effort to develop portions of products that do not differ and limiting the effort needed to derive and evaluate the best fit of alternatives for portions that do differ. As needed capabilities change over time, the platform is modified to support building products having those capabilities.
- An envisioned product can be realized and explored in a "pure" idealized expression of its behavior in software. A software-only realization can be built to operate in an actual or facsimile ecosystem. The decision to realize any behavior in hardware form is made either to enable interactions with a physical ecosystem or to gain the improved performance or reliability of a hardware realization.
- The behavior that is to be realized through the physical mechanisms of a hardware device can be expressed in software. A device can be built or selected to exhibit such software-specified behavior. Software can be used to specify the behavior of a device as requirements for its fabrication. Software can be embedded in a device to functionally enhance and flexibly adjust, monitor, or modify or transparently replace a device over time as needs and operational circumstances warrant.
- A device may be virtualized in a software-realized surrogate that emulates the behavior of an otherwise inaccessible device (e.g., prior to manufacture, not connected for access, or having failed in operation), effecting services or obtaining data via other devices. Software can simulate a device's approximate internal functioning for the purpose of characterizing its properties and sensitivity to external influences and to compare how variations in its modeled operation would cause its behavior to differ.
- A hardware device can be software enabled either on the device itself or within a software component that encapsulates access to the device's services. Similarly, software can encapsulate access to any entity to customize, enhance, or enable

changes in its apparent services or behavior. Conversely, critical software capabilities can be implemented in hardware to ensure attainment of specific quality criteria (e.g., enhanced responsiveness, accuracy, or security).