White Paper

PTC®

Systems Engineering: A Critical Practice for Successful Delivery of Smart, Connected Products

Products in nearly every industry today are quickly becoming smart, connected components of larger software-intensive systems and systems of systems. The practice of systems engineering must adapt to address the challenges that come with this level of complexity and to overcome barriers to success.

The Capabilities of Smart, Connected Products

Smart products have driven significant societal changes in the past decade, saving lives, enabling instant global communication and access to unlimited information, making travel safer, changing the way warfighters engage the enemy, and enabling consumer products to do more than anyone ever imagined. Along the way, discrete-manufactured products have become a highly complex system of systems in which products are not only smart, but connected.

This new wave of innovation has unleashed products that are interdependent of each other and supporting ecosystems. These systems offer new types of functionality that remotely perform defect repairs, add new features, analyze sensor data, change product configuration, automatically schedule service, deliver wide-ranging information to the user, and more. The potential for these devices is expanding as quickly as the physical number of smart, connected products.

Smart, connected products are all around us

Evidence of systems thinking in products and services began to influence our personal lives years ago with the advent of complex systems of systems like the global telephone network and cellular phones that were always on and continuously connected to the carrier's network. The paradigm has now evolved to the point that the mobile device itself is treated as simply a platform for continuous delivery of customer value in the form of applications, services, and information. Even hardware design flaws can be seamlessly resolved through delivery of software updates.

The latest generation of automobiles has already been transformed through the use of numerous advanced sensors, 'drive-by-wire' technologies, and increasing use of software throughout the vehicle. Automakers are now in the process of transforming the latest generation of stand-alone vehicles into systems of systems by connecting them to information and service networks. This approach is enabling capabilities that could never be realized with disconnected vehicles, no matter how sophisticated.

For instance, vehicles that provide automated detection and notification of service needs are already available to consumers. This approach will soon deliver even more significant capabilities, like centralized remote control of braking and acceleration on vehicles in heavy-traffic areas to optimize traffic flow. And work is already under way to produce computer-piloted vehicles that combine classic autonomous control with optimal point-to-point navigation by taking advantage of persistent communication links with centralized traffic management systems and other Internet-based information sources.

Even farming and construction machinery is not immune to this technological shift. Grain harvesters are being transformed into a sophisticated system of systems and are now equipped with sensors, cellular and satellite-based data communication links, and an increasing volume of software. They communicate with Internet-based supporting software systems, which enable the coordination of services needed to keep the machine in the field and operating at maximum efficiency at all times. During peak harvesting season when downtime is costly, every machine in the fleet can be monitored in real-time, with support software even predicting machine maintenance needs before they occur.

The introduction of smart, connected products has given manufacturers countless ways to add value without adding proportional cost to products and thus recover greater profit from the sale of each unit. More importantly, it has enabled the introduction of new, recurring revenue streams that are based on services wrapped around the core product. This service-centric product model, or product servitization, gives manufacturers the freedom to amortize most or all of the product's engineering and manufacturing costs over the product's service life. It also brings an end to the notion of 'planned obsolescence,' since the physical product is now a platform for the delivery of higher-value services. In its place, we will see the notion of designing for products that evolve over time.[1] This is part of a larger fundamental shift currently under way in the product manufacturing paradigm sometimes called the third industrial revolution.[2] Like the first and second industrial revolutions, it will change our world dramatically.

Smart, connected products drive complexity

In addition to great opportunity for innovation, this paradigm shift is also creating a paradox for manufacturers. Such capabilities naturally require greater complexity in all facets of the product lifecycle. Beginning with the first industrial revolution over a century ago, manufactured products have shifted from mechanical to electro-mechanical to mechatronic to software-dependent designs. These transitions enabled manufacturers to innovate rapidly while lowering costs, improving quality, streamlining compliance, and delivering products to market faster.

The same transitions have introduced dramatic increases in complexity, which is at the root of the paradox because product complexity stands in the way of achieving those goals. More complex products require a much deeper understanding of needs, risks, and requirements before designing more complex product designs, more complex manufacturing processes, and more complex support processes during the service life of the product. All this adds up to greater cost, lower quality, less compliance, and longer development cycles, which work together to inhibit innovation.

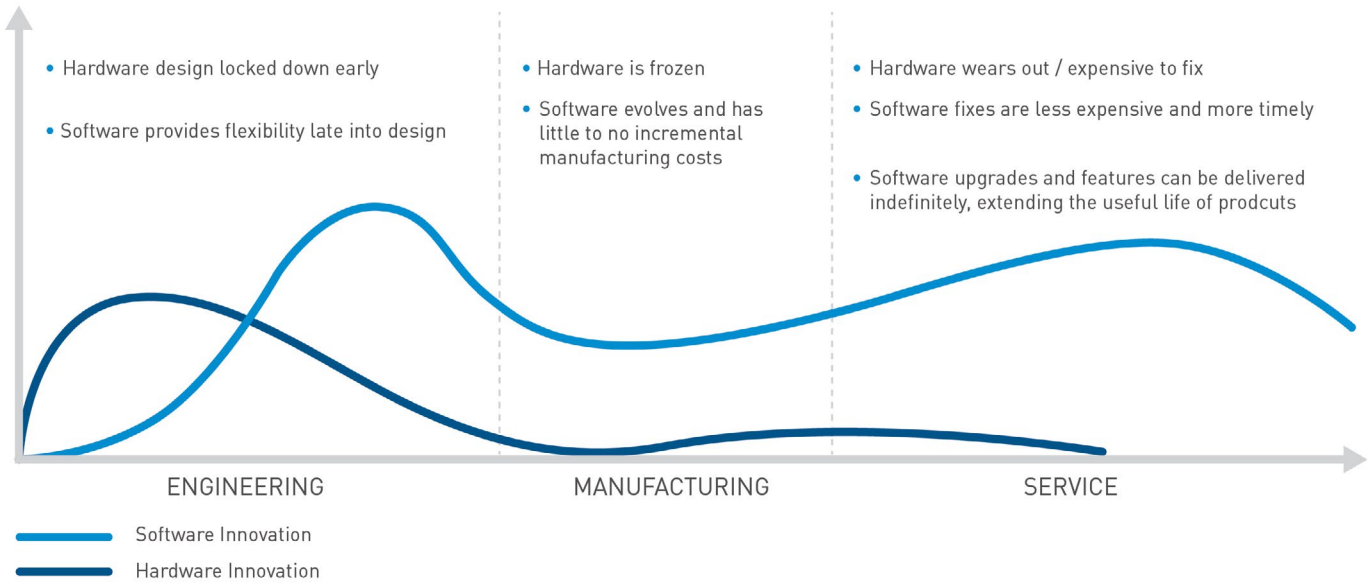
The most recent shift from mechatronic to software-dependent product designs has not only introduced another dramatic increase in product complexity, but also a different kind of complexity that is related to the unique qualities of software. One of those unique qualities lies in the fact that a seemingly insignificant change to one line of software code can lead to unanticipated and unwanted behavior in a different part of the program that appears to have no connection to the portion of the program where the change was made.

This kind of software defect—sometimes referred to as a side effect—illustrates a type of behavior in software-dependent products that appears to be unpredictable or even random. In reality, the side effect is not true random behavior but rather the result of the highly complex nature of software programs. Software-driven complexity is still not widely understood to the degree necessary to anticipate and manage the associated risks effectively.

Finally, the current transition from stand-alone, software-dependent products to smart, connected products that compose systems of systems is introducing another new type of complexity. Consequently, manufacturers are facing new challenges that require significant changes and advances in their traditional product engineering, manufacturing, and service processes.

PTC Lets You Leverage Software for Continuous Innovation

While hardware degrades, software only gets better



The Challenges that Product Complexity Bring to Engineering

Smart, connected products bring complexity challenges not only to design and engineering, but also to the product development process itself. One way to categorize this complexity is through the lens of the classical engineering triple constraint of cost, schedule, and performance.[3]

Increased development cost and longer time-to-market

It simply costs more to specify, design, and build a more complex product because the effort required is proportional to the level of complexity. In fact, it is more accurate to say that the effort required scales *disproportionately* to the level of complexity. Each new feature, behavior, or non-functional capability not only adds its own additional complexity, but often affects the existing product design as well, adding more complexity, effort, and cost. At the same time, the additional effort and cost typically lead to trade-off decisions that result in adding time to product engineering and manufacturing schedules.

Greater risk, lower quality

The cost of product liability and regulatory compliance is ratcheting up rapidly. Increasing complexity in products makes it even more difficult to ensure the product will function as intended regardless of the way it is used, the environment in which it is used, and the length of time it is in service. In today's rapidly escalating liability and regulatory environment, manufacturers face huge risks if one of their products fails in a catastrophic way. This, in turn, forces them to pour additional effort into the identification, management, and mitigation of risk, as well as more expensive quality assurance, verification, and testing processes.

Barriers to innovation

Good ideas for new products and enhancements to existing products are often deferred due to concerns about cost, schedule, and risk that arise from the added complexity required to implement them. Further, engineers distracted by existing complexities in products have less time and energy to focus on innovation. Consequently, they tend to shelve a new idea before it can be vetted.

New kinds of complexity

As sophisticated and highly complex discrete products become part of even larger systems of interconnected products, the level of complexity not only increases but takes on new characteristics. Systems engineering originally arose to address the complexity of monolithic systems. As a system of systems exhibits new complex behaviors, traditional systems engineering practices must be revised to address them.

Increasing use of software

The final challenge driving change in systems engineering is the explosive growth of software in manufactured products. Software-intensive products can do more—better, faster, cheaper, and with greater reliability—than mechatronic products of roughly equal capability.

The figure below shows how software-intensive products can give product manufacturers significant advantages over mechatronic products.

Those advantages, however, come with a mirrored set of challenges that, when not managed, can undo all the gains. Organizations that are unaccustomed to managing these profound differences are often surprised by unwanted and costly consequences. For example, if an auto manufacturer pushes a software change to all affected vehicle models in the field remotely and that change includes a new defect that results in the failure of a safety-critical sub-system, the resulting consequences could be widespread, devastating, and costly.

Software brings new forms of capabilities and complexity, and these advantages and challenges stem from the fact that software is different than hardware in its very nature. As a physical entity, hardware typically adheres to binary rules of functionality; it either works or it doesn't. At the root of it, software is not a physical entity. Instead, it is fundamentally a sequence of instructions executed by a microprocessor that is embedded in a product.

Software is inherently more difficult to understand, especially the side effects of one line of code on another—and even more so when the two lines of code may be in different software modules that are part of separate hardware components. As a result:

- Software development *effort* is harder to plan, estimate, and measure
- Software-driven *risk* is harder to identify and mitigate
- Software *defects* are harder to anticipate and find, and thus *quality* is harder to assure and measure
- Software *change* is harder to plan, estimate, and measure
- Software *change* occurs at a much greater frequency than hardware changes, making it difficult to manage using conventional engineering change management processes and tools

As smart, connected products are increasingly defined by software, their different nature is driving significant changes in processes and tools across the entire product lifecycle, including engineering, manufacturing, and service. In short, the rapid introduction of software into high-technology products has created a completely new world for product manufacturers filled with monumental opportunities and challenges.

Software is the key to innovation in smart, connected products

> Software-driven innovation is the new frontier of competition, helping companies drive more rapid innovation and creating opportunity due to the increased flexibility and agility of software-based products."

Jim Brown

Issue in Focus: Systems and Software-Driven Innovation - Complexity and Opportunity in the Mechatronic Era, Tech-Clarity, Inc., 2011.

Other Factors Driving Changes in Systems Engineering

While product complexity has played a predominant role in driving changes to the systems engineering discipline, other forces have also come into play that are likewise driving changes to the practice of systems engineering.

Global competition, customer bases, and development: 'Better, faster, cheaper'

Product manufacturers have begun to extend the reach of their products and resulting revenues to global markets. This has naturally driven more competition, as new competitors are constantly entering any given local or regional marketplace, yet are also able to reach the same global marketplace.

Consumers and manufacturers have clearly benefited from the resulting innovation, lower product cost, and faster time-to-market, but there is another side to this matter. The relentless drive for 'better, faster, cheaper' also places enormous pressure on manufacturers to change their methods of bringing products to market. Many of the business, engineering, manufacturing, and service processes still widely in use today are crumbling under the strain and failing to scale to meet the need.

Exploding liability and regulatory burdens

The costs associated with constantly increasing product liability and mounting regulatory burdens have become a primary concern of product manufacturers in recent years, often driving what appears to be irrational behavior when seen from the outside. The risks associated with this trend are potentially monumental and catastrophic, with only limited means of managing and mitigating those risks in advance. To address these issues, manufacturers know they must improve product quality, safety, and reliability while complying with the growing and increasingly conflicting demands of government regulators and industry standards.

To achieve these improvements in product quality, safety, and reliability while at the same time relying on increasing complexity to make products better, faster, and cheaper, manufacturers are forced to rethink the way they do business across the entire product lifecycle. A small but important advantage in the midst of this difficult scenario is the fact that the changes needed to address this challenge are nearly identical with those needed to address the 'better, faster, cheaper' challenge, though the common changes must be incorporated carefully to realize the dual benefit.

Rapid advances in technology

Recent technological advances in critical areas affecting product manufacturing (e.g., materials science and Nano technologies, sensors, and wireless connectivity,) are affecting manufacturers and systems engineers in profound ways. These advances are enabling game-changing product innovations at a rapid rate and each individual advance requires adaptation of practices, methods, and tools across all engineering disciplines. Of course, such changes within each discipline require adaptation in related disciplines as well. All this adds up to a lot of pressure on systems engineers to adapt to each new technological change rapidly and effectively so as to take full advantage of it.

Practice Requirements for System Engineering

Systems engineering adds more value than ever before

> Better/more systems engineering correlates to shorter schedules by 40 percent or more, even in the face of greater complexity."

Jim Brown

Issue in Focus: Systems and Software-Driven Innovation - Complexity and Opportunity in the Mechatronic Era, Tech-Clarity, Inc., 2011.

Most engineering organizations today still practice the classical model of separate engineering teams organized by discipline, each working primarily in its own silo. Systems engineers throw sub-system, component, and interface specifications 'over the fence' to mechanical, electrical, and software teams. Those teams work primarily on their own to deliver their respective designs and code, which are not integrated and tested at the system level until late in Most engineering organizations today still practice the classical model of separate engineering teams organized by discipline, each working primarily in its own silo. Systems engineers throw sub-system, component, and interface specifications 'over the fence' to mechanical, electrical, and software teams. Those teams work primarily on their own to deliver their respective designs and code, which are not integrated and tested at the system level until late in the engineering cycle. This creates late development cycle rework and causes an expensive and time-consuming test-fix cycle.

These challenges thwart organizations' ability to deliver innovative, smarter, and more reliable products to market faster and cheaper. The classic 'divide-and-conquer' engineering strategy simply cannot address these holistic, crosscutting challenges.

Products and systems crossed a complexity threshold many years ago, making classic waterfall systems engineering practices inadequate for identifying and driving out the most critical risks early enough in the lifecycle, thereby requiring late rework and leading to significant cost overruns and delayed market release.

It can even lead to loss of market share and customer confidence. Thus a multidisciplinary, iterative systems engineering approach to product design is imperative if manufacturers are to gain control of this complexity.

Systems engineering must be iterative

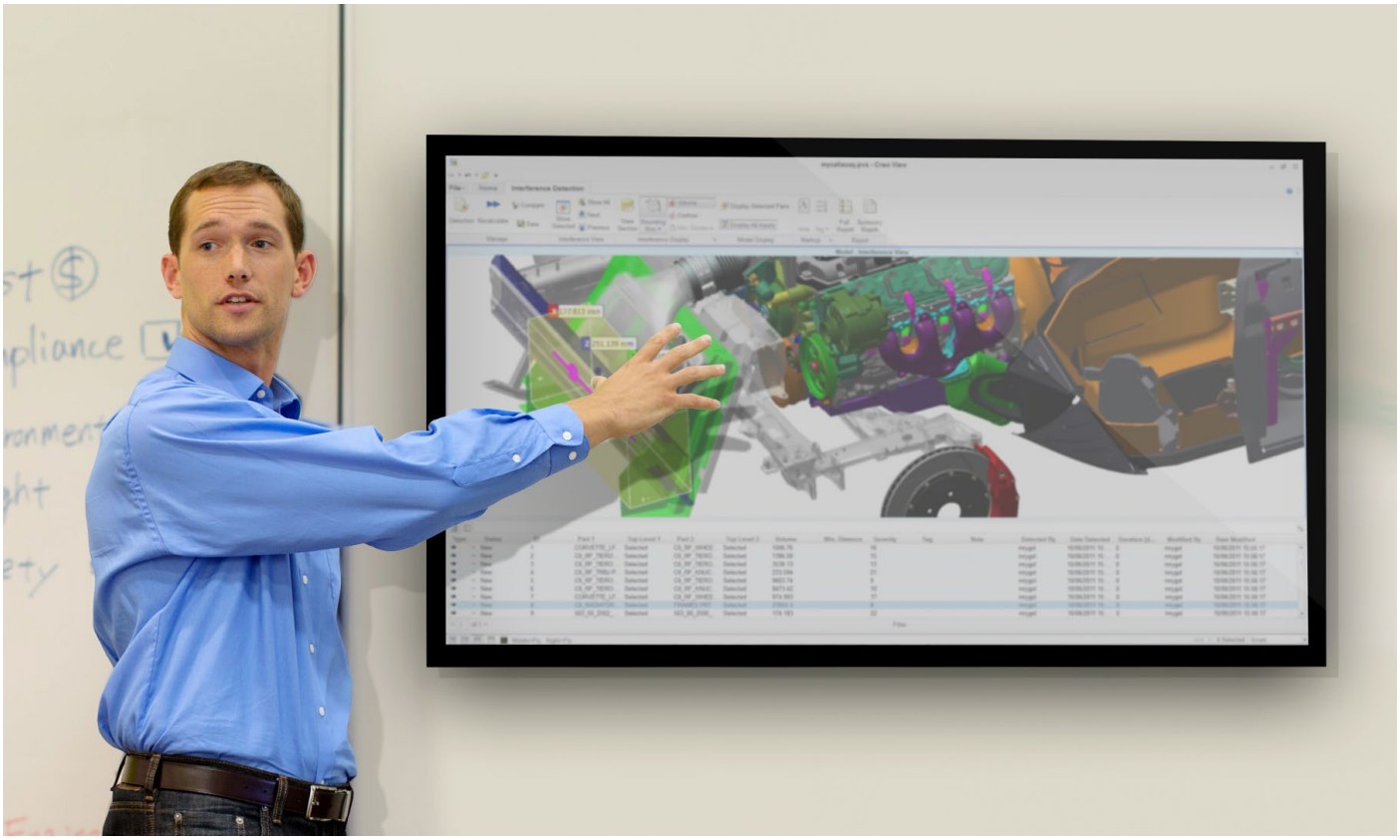
As previously stated, the greater the complexity of the system under consideration, the less likely manufacturers will be able to drive out critical risk early enough to efficiently and effectively design and build the product right on the first try. By iterating through early steps in the lifecycle—starting with systems requirements and architecture and continuing through component design and verification—engineers can 'learn what they don't know' and adapt system requirements and architecture accordingly.

Systems engineering must be highly collaborative

Since the biggest challenges in building a complex system of systems are crosscutting and holistic in nature, it no longer makes sense to have engineers organized into teams by discipline. Instead, manufacturers are learning they must reorganize into smaller, multidisciplinary teams in which all key engineering disciplines are represented—usually including systems, mechanical, electrical/electronic, software, manufacturing, and service engineers. In this environment, multidisciplinary collaboration is not only encouraged through proximity but must be enabled by agile processes and advanced collaboration technologies. Only in such an environment can the most critical issues associated with complex products be identified and mitigated early in the lifecycle.

Systems engineering must address the new product lifecycle

The complexity of today's smarter products demands an even more holistic product lifecycle approach that begins when the initial needs for a new product



are defined and continues until the product is finally retired from use. User and usage data can now be harnessed from products after they have been delivered to customers. In addition, the paradigm shift from the discrete product as the primary source of revenue to smart, connected products and servitization as primary revenue stream means that systems engineers must not only focus more on the service part of the product lifecycle and harness captured usage data, but also adapt to seeing the lifecycle in a whole new way.

Systems engineering must validate requirements early and often

The process of requirements validation—or determining that the requirements are correct to ensure that the right product will be built—must adapt so that system requirements are more thoroughly validated much earlier in the product engineering lifecycle. The effects of numerous methods and tools must be realized to accomplish this goal, including expanding the use of behavioral modeling and simulation during the system requirements and architecture phases of the lifecycle.

Systems engineering must be model-centric

Specifically, modeling and simulation must take on a new role in the practice of systems engineering. Text and static diagrams have been the primary and central tools of specification and communication for the systems engineer for decades, but the complex nature of smart, connected products and systems is driving a shift toward models. In general, models are a less ambiguous and more holistic tool of specification and communication in engineering.

Model-based systems engineering (MBSE) is merging with system simulation languages to form an integral means of capturing and validating all aspects of system requirements and architecture in one seamless model. This transition could potentially transform the entire systems engineering practice. For instance:

- SysML, along with simulation model notations, is now being used by many engineering organizations to replace some or all of the textual and diagrammatic specifications

- System modeling has advanced to the point that systems engineers may soon be able to seamlessly and automatically transform system requirements models into system architecture models and, in turn, into high-level hardware and software design models
- System models are increasingly being used to drive system behavioral, algorithmic, and parametric simulation, enabling quicker and less ambiguous validation of product needs, system concept, and system requirements

Models are powerful and hold much promise, but they must be managed through the lifecycle like other artifacts. Change and configuration management, traceability, and systematic reuse are all lifecycle management concepts that must be uniformly applied to models to maximize the value of models.

Systems and software engineering must work together seamlessly

As previously shown, software is not like hardware, so software complexity and change are not like hardware complexity and change. It is interesting to note that systems and software engineering are more alike in many ways than systems and hardware engineering, and this parallel nature has not gone unnoticed by leaders in the systems and software engineering professions.

The close relationship between systems and software engineering is not merely coincidental. It is an outgrowth of the fact that software, by its unique nature, lends itself to being treated as a system in nearly every facet (architecture, design, modeling, simulation, development, testing, etc.)Therefore, as software continues to grow in volume and role in products, the value of the high correlation between systems and software engineering methods increases as well and leads systems and software engineers to share and adapt each other's methods and tools.

Systems Engineering of Smart, Connected Products

Innovative systems of smart, connected products are creating unprecedented capabilities and complexity. Designing for systems of systems is the new reality for manufacturers. The stakes have been raised as manufacturers face pressures from global competitors, regulatory risks, and accelerating innovation.

Systems engineering can help manufacturers navigate this new terrain. Effective systems engineering provides an inclusive, collaborative solution that enables model-driven design, early simulation and testing against requirements, and integration between systems and software engineering.

PTC understands today's smart, connected products demand a systems engineering practice that is more robust, collaborative, and model-driven. Systems engineers must harness technology solutions that work across multi-disciplines along the entire product lifecycle, capturing from cradle-to-grave data for validation and verification.

To learn more about how to address systems engineering challenges facing manufacturers and how leaders are transforming to meet those challenges, visit PTC.com/topics/systems-engineering.

For more in-depth information about the integrated family of systems engineering technology solutions that PTC delivers, visit: PTC.com/solutions/all/systems-engineering.

[1] The New Manufacturing Lifecycle: An Interview With PTC's Hepplemann, Forbes Magazine, April 3, 2013, www.forbes.com/sites/maribellopez/2013/04/03/the-new-manufacturing-lifecycle-an-interview-with-ptcs-hepplemann.

[2] Special report: Manufacturing and innovation – A third industrial revolution, The Economist, April 21, 2012, www.economist.com/node/21552901.

[3] In this context, the term 'performance' includes all product needs and requirements, both functional and non-functional.

J4853 –Systems Engineering–WP–EN–0115