Tech-Clarity

Engineering Buyer's Guide for Multi-Discipline Systems

The Expert Solutions Guide

Michelle Boucher, Vice President, Tech-Clarity

Table of Contents

Executive Overview

Fierce global competition continues to drive companies to seek new ways to competitively differentiate their products. Many achieve this differentiation through smarter and more connected products. This approach creates many innovative possibilities for new products and services.

This buyer's guide explores the capabilities needed in a complete systems engineer solution.

While smarter products and connectivity create exciting opportunities for innovation, they also bring unique challenges.

While smarter products and connectivity create exciting opportunities for innovation, they also bring unique challenges and add new levels of complexity. Much of this complexity comes from requirements to integrate mechanical components, electronics, and software. Connectivity adds further complexity as you add sensors, streaming data, and an ecosystem of connected systems. Whether you are a systems engineer, product architect, or domain-specific engineer, addressing this complexity requires expert systems engineering practices.

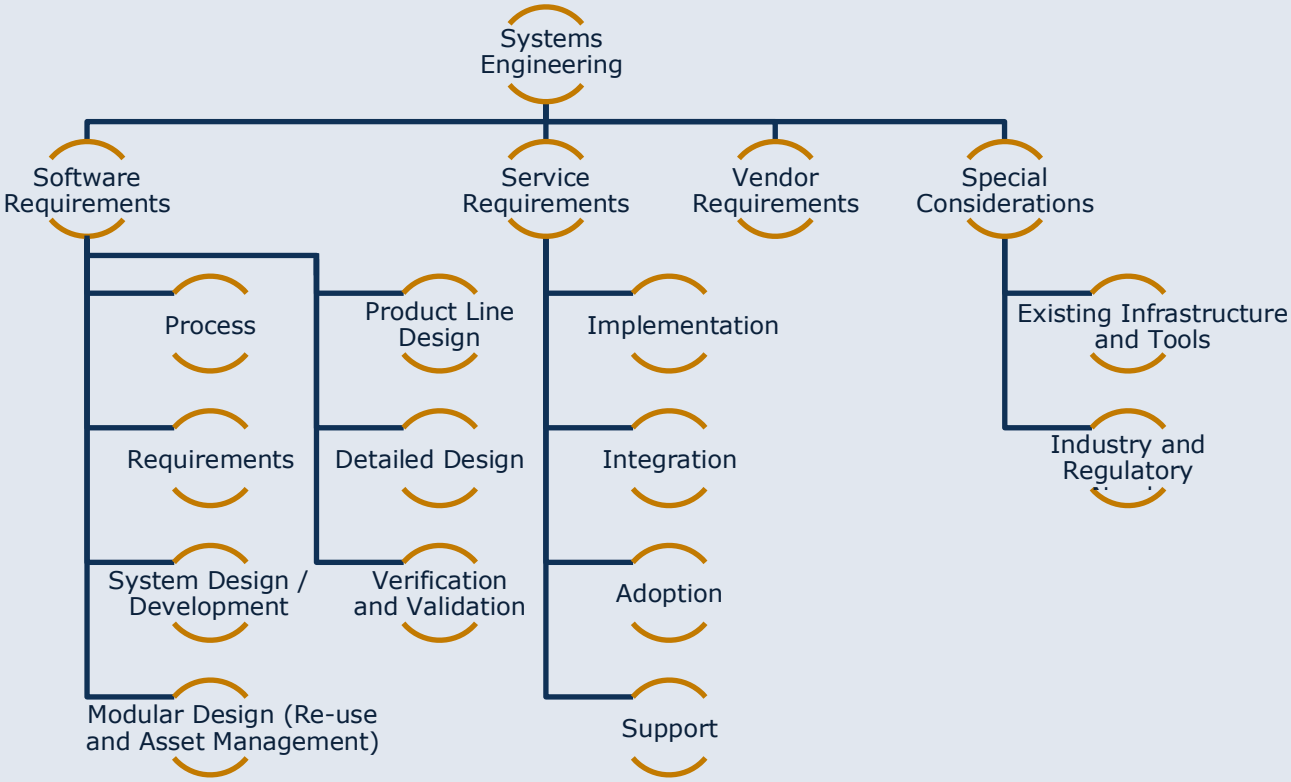
To set the foundation for expert systems engineering practices, companies should focus on the entire systems engineering process, ensuring there are solutions for all aspects of systems engineering. Expert systems engineering practices will help companies become even more competitive in ways that will lead to higher growth and greater profitability.

The right technology is an essential part of implementing and supporting these expert systems engineering practices. The right technology will enable improved collaboration, better traceability with a digital thread, and earlier visibility to potential problems. This will result in a more efficient development process while reducing the risk of finding errors late in the process, helping to avoid delays and increased costs. This buyer's guide will help manufacturers select the right software to support systems engineering.

To set the foundation for expert systems engineering practices, companies should focus on the entire systems engineering process, ensuring there are solutions for all aspects of systems engineering.

This guide comprises four major sections covering systems engineering software tool functionality, service requirements, vendor attributes, and special company considerations (Figure 1). Each section includes a checklist of key requirements to support your selection process for systems engineering software tools.

FIGURE 1: SYSTEMS ENGINEERING EVALUATION FRAMEWORK



This framework will be useful to a variety of roles, not just systems engineers. As companies develop smart connected devices and products, many will struggle with the challenges solved by expert systems engineering practices. Newer roles will also emerge that involve end-to-end responsibility for designing and architecting the overall system, and mapping business needs to the system and technical requirements. Functions also include developing the technical specifications for the connected system. These roles include, but are not limited to titles such as:

- IoT Solutions Architect
- DevOps IoT
- Internet of Things Solutions Engineer
- M2M and IoT Product Management and Engineering
- Technology Manager, Software Engineering (IoT)
- Solutions Engineer (IoT)
- Principal Technical Architect (IoT)
- Innovation Manager for Industry 4.0
- Product Manager

This guide is not an all-encompassing requirements list, but provides a high-level overview of systems engineering needs. In addition to the systems engineering capabilities discussed in this buyer's guide, companies should also consider individualized needs for the tools for each engineering discipline.

The Business Value of Systems Engineering

Past research from Tech-Clarity has shown the importance of expert systems engineering processes. One of the drivers for expert systems engineering practices is the increase in embedded software. Tech-Clarity's *Developing Software-Intensive Products: Addressing the Innovation Complexity Conundrum* found that 75% of surveyed discrete manufacturers would increase the amount of software in their products.

75% of surveyed discrete manufacturers plan to increase the amount of software in their products.

This growth is due to the strategic ways embedded software helps them achieve their growth and profitability goals. Survey respondents report the following top product strategies driving the use of software:

- Improve product capabilities (74%)
- Develop smarter products (68%)
- Increase product innovation (62%)
- Enhance ability to tailor products (49%)
- Enable platform design (43%)

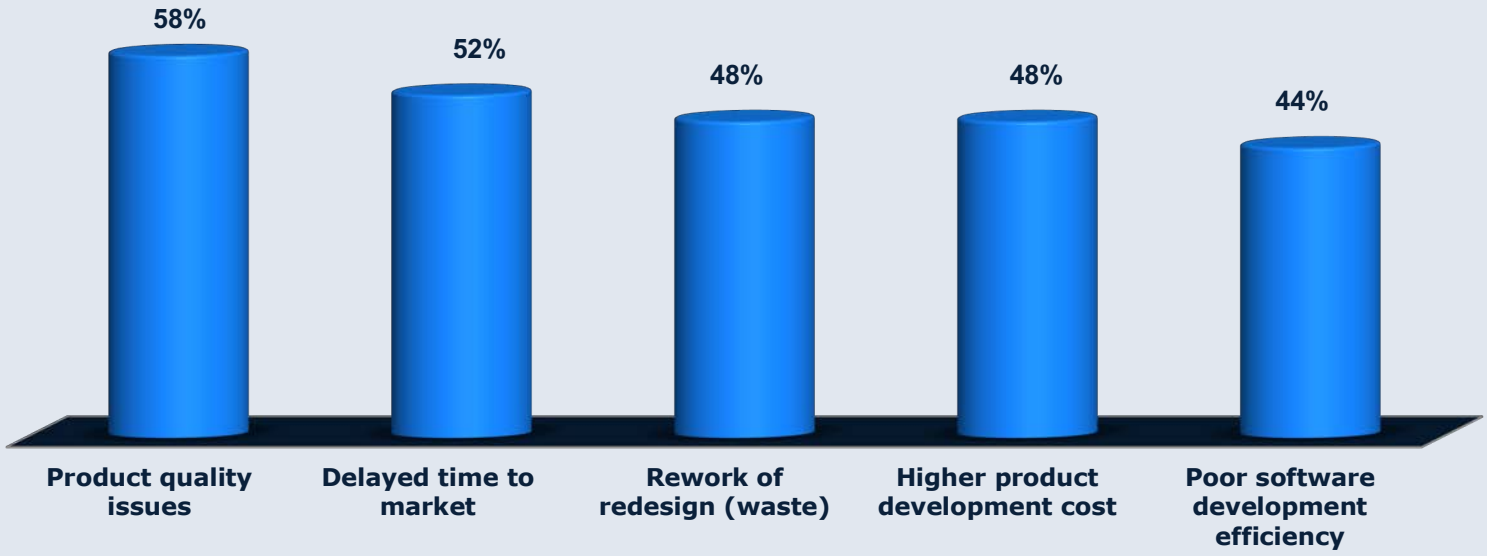
All of these product strategies differentiate products in important ways and help improve competitiveness. However, manufacturers find that when they do not have expert systems engineering practices, they undercut the business goals driving the need for embedded software in the first place.

Manufacturers find that the lack of expert systems engineering practices undercut the business goals driving the need for embedded software.

Figure 2 (next page) shows the negative business impacts when developing products with embedded software.

By following expert systems engineering practices, companies have a framework by which to manage systems development. Systems development is so complex, that without these practices, many things can go wrong. Missing these problems hurts product quality. Poor quality tarnishes brand reputation and a likely loss of customers, especially when there is so much competition for those customers. Fixing problems can cause delays, particularly when they are found later in the development cycle. Delays mean late to market, which leaves less time to bring in product revenue and capture market share. In addition to delays, required rework and redesign to correct problems also adds cost. Tracing the root cause of problems and fixing them takes time away from new design and development work, driving up cost and hurting efficiency. All of these issues lead to less revenue and higher cost, hurting profitability.

FIGURE 2: NEGATIVE BUSINESS IMPACTS DEVELOPING PRODUCTS WITH EMBEDDED SOFTWARE



The good news is that manufacturers can avoid these negative business impacts by implementing expert systems engineering practices. Technology plays a significant role in enabling expert systems engineering practices. It can also help manufacturers address the complexity, from getting the requirements right to managing the product lines needed to appeal to various customer segments.

The good news is that manufacturers can avoid these negative business impacts by implementing expert systems engineering practices.

Start with Process Definition

Defining the process is a logical place to start when implementing expert systems engineering practices. Tech-Clarity's *Developing Software-Intensive Products: Addressing the Innovation Complexity Conundrum* finds that a top challenge of systems engineering is managing complex global development networks, reported by 49% of manufacturers. Defining the process helps address some of this complexity.

Clearly defined processes improve communication and productivity. Defining processes also helps identify those practices that work

49% of manufacturers report that managing complex global development networks is a top challenge of systems engineering.

best and ensures the team follows them consistently. Deploying consistent best practices will lead to a more efficient development team, which means getting to market sooner.

Defining processes helps identify those practices that work best and ensures the team follows them consistently.

Improved time to market provides a competitive advantage and optimizes the window of opportunity to recoup development investments, both of which will improve profitability.

The role of technology to support the process is:

- Streamline and simplify process authoring
- Facilitate process adoption
- Provide governance to enforce processes
- Capture metrics to assess process effectiveness
- Enable continuous improvement
- Manage process and task ownership
- Integrate with systems engineering deliverables

The following table lists software requirements to support this:

FIGURE 3: FUNCTIONAL REQUIREMENTS FOR PROCESS DEFINITION

REQUIREMENT	CONSIDERATION
Authoring tools to capture best practices	Documenting best practices provides a consistency that will lead to greater efficiency. The authoring tool should simplify the documentation process and automate as much as possible.
Graphical modeling of business processes and best practices	Graphical models provide a visual framework to communicate complex processes more easily. With improved understanding, processes are more likely to be followed. Consider a tool that is compliant with an industry standard such as OMG BPMN (Object Management Group Business Process Modeling Notation), which is royalty-free and provides a proven standard based on industry thought leadership.
Automated best practices through automatic notifications and prompts for next steps	Process automation makes the processes easier to implement and follow, which leads to better adoption. Functions such as automated to-do lists and email notifications streamline handoffs, helping to overcome process bottlenecks and improve efficiency.

REQUIREMENT (CONTINUED)	CONSIDERATION
Process modifications	The ability to easily make changes leads to continuous improvement by applying lessons learned. This leads to efficiency and also captures processes that will lead to higher quality and lower cost. Processes should be modifiable by process authors, practitioners, and those closest to systems engineering processes, such as SysML modelers. SysML is a modeling language for systems engineering. It was based on UML, which is a software-focused modeling language.
Sub-process changes automatically applied to all other processes that use that sub-process	This leads to greater process consistency. It also saves time for process authors by minimizing manual updates. In addition, it supports continuous improvement to apply and take advantage of lessons learned consistently.
Library of existing processes available to define new processes or project-specific processes	A library will save time by reusing existing proven processes rather than creating them from scratch. Also, best practices learned from previous projects can be applied to new ones.
Project-specific processes based on existing best practices	To save time, existing best practices should be copied to create project-specific processes. Processes should be editable for individual project needs too. This ensures new project plans are based on the latest best practices, while providing flexibility to accommodate specific project needs.
Import and export process content to commonly used formats	Common formats such as Microsoft Word, PDF, XML, and Microsoft Project plans should be supported. Support for formats used for existing process definitions makes it easier to take advantage of them, saving time. It also improves adoption because project plans can be defined in tools of choice.
Automated auditing	This ensures processes are followed as defined.

Manage Requirements

Requirements establish the foundation for a product. They guide development activities. The ideal requirements management solution needs to manage requirements from the high-level system view to the individual component. Two of the top systems engineering challenges are managing change, reported by 56% of the respondents to Tech-Clarity's survey, and closing the loop from requirements to validation, reported by 48%.[1] A requirements management solution can address this with requirements traceability so that the dependencies across requirements are understood when making a change. Traceability makes it possible to trace a requirement from its definition to its deliverable to its test, across all engineering disciplines.

The ideal requirements management solution should manage requirements from the high-level system view to the individual component.

The following list details key criteria for a requirements management solution:

FIGURE 4: FUNCTIONAL REQUIREMENTS FOR MANAGING REQUIREMENTS

REQUIREMENT	CONSIDERATION
Definitions for both top-level and sub-system requirements	Managing requirement hierarchies helps manage system complexity. This makes it easier to maintain product quality while keeping costs low and streamlining compliance. Requirements created elsewhere should be importable.
Support for requirements across all disciplines, including mechanical, electrical, software, and others	A single source of requirements across all disciplines reduces the chance of errors and improves communication with a consistent definition of the requirements.
Traceability and interdependencies across requirements, design, implementation, and test	Traceability across requirements, design, implementation, and test is necessary to properly understand the impact of changes and ensure all affected components are updated. Traceability is also needed for regulatory compliance.

REQUIREMENT (CONTINUED)	CONSIDERATION
Requirements reuse	Reusing requirements saves time. Consider how sophisticated the relationships between reused requirements should be. For example, when reusing a requirement, should there be a link to the original requirement? When changing a requirement, should that requirement be flagged everywhere it was reused?
Requirements diagrams	Visual models such as SysML provide a visual framework for requirements, making it easier to understand the relationships across requirements.
Requirements approvals	Companies that must comply with regulations such as FDA 21CFR should consider solutions that offer the needed audit trail, complaint e-signatures, and other functionality that will support compliance requirements.
Requirements allocated to every element of the system model	Mapping requirements to the system model helps manage complexity by visually managing interdependencies across requirements, subsystems, and components.
System model filtering	Filtering allows you to select a model element and identify all associated elements, supporting traceability. Traceability reduces the chance of errors when implementing changes, which can reduce cost overruns and improves quality.
Change impact analysis	When changing a model element or requirement, the tool should identify other impacted model elements. This saves the time to execute change orders and makes it easier to understand the full implications of a change.
Integration with other requirement management tools	This allows the flexibility to take advantage of investments in existing tools that are working well within an organization.

REQUIREMENT (CONTINUED)	CONSIDERATION
Integration between requirements and other design and systems engineering tools	Integration between requirements and other systems design tools improves traceability, creating a digital thread. Support for Open Services for Lifecycle Collaboration (OSLC) offers an open mechanism to link data across domains, applications, and organizations.

Design the System

The root cause of many systems engineering challenges is complexity. A system model can help to manage complexity. For one, it provides a visual representation of the system. The visual reference supports the design, analysis, verification, and validation of the system. It also provides a common reference for the system that is easy to understand and thus improves communication and collaboration across the development team. Improved collaboration leads to a more productive and efficient team.

A system model provides a common reference that is easily understood and thus improves communication and collaboration across the team.

The following is a list of software requirements to support system modeling:

FIGURE 5: FUNCTIONAL REQUIREMENTS FOR SYSTEM DESIGN

REQUIREMENT	CONSIDERATION
Compliance with industry standards	A method to consider for system modeling is SysML. SysML is a modeling language for systems engineering.
SysML executable behavior models	SysML executable behavior models can simulate system behavior for early verification. This early visibility makes it easier to determine if the system design will meet customer needs.

REQUIREMENT (CONTINUED)	CONSIDERATION
Parametric diagrams	Parametric diagrams describe constraints on system properties. They are used for performance and quantitative analysis, which leads to higher quality.
Parametric diagrams integrated with simulation/evaluation tool	Integrating the parametric diagram with a tool such as Matlab or Modelica allows the tools to work together for early validation, optimization, and trade-off analysis. This can lead to higher product quality in less time.
Integration with requirements management tools	Integrating requirements and the system model helps manage complexity by providing a visual way to manage interdependencies across requirements, subsystems, and components. It also maintains traceability from the system model across requirements.

Design the System to be Modular

As found in Tech-Clarity's research,[2] 42% of manufacturers report that design reuse is one of their top systems engineering challenges. Reuse helps to reduce time to market by reducing design, development, and test time. In addition, with complex systems, reusing proven and validated components and subsystems means fewer errors.

With complex systems, reusing proven and validated components and subsystems means fewer errors.

The challenge is that systems are so complex; it is hard to identify reusable components. A modular design strategy can help.

A modular system breaks the system down into smaller sections that are easier to understand. These smaller sections are also easier to validate as the design evolves, so you catch problems sooner. After validation, components and subsystems can be published to a library of reusable assets for future use in similar systems.

The following is a list of functions to support modular system development:

FIGURE 6: FUNCTIONAL REQUIREMENTS FOR MODULAR DESIGN

REQUIREMENT	CONSIDERATION
Library of reusable subsystems and components	A repository of existing components facilitates reuse, saving time. Using existing validated components reduces the risk of errors, resulting in higher quality and less cost. Consider a solution that complies with OMG RAS, an industry specification that provides guidelines on industry best practices for reuse.
Library components define interfaces, variants, and requirements	Library components with these embedded definitions make it easier to properly reuse components in future systems.
Search tools to find reusable components or subsystems (assets)	Functions such as indexing, classification, and tagging help make it easier to quickly find components that can be reused, saving both design and test time.
Library publishing tools	The ability to publish subsystems, interfaces, components, use cases, and requirements from models and interface files enables greater reuse. This can also support a top-down modeling or a bottom-up system mining approach.
Version management and history of reusable assets	A version history makes it possible to notify others who have used a component when a new version is available. System models should also manage the component's version.
Dashboards to measure reuse	Metrics help to identify what's working well and what is not. Tracking how often components and subsystems are reused measures the effectiveness of the reuse strategy and enables continuous improvement of the library.

Support Product Line Variants

From a system perspective, a product line is a group of similar products. A product line consists of different variants, each with common base features as well as variable options. The goal is to create a common platform with reusable components that can be put together to create unique product models. Enabling platform design is one of the top reasons companies add more software to their products, yet 49% report designing product platform variants as a top challenge.[3] The ability to tailor products to specific customer needs makes products more desirable and therefore more competitive. Software provides a cost-effective method for offering different product functions. This strategy becomes even more powerful as companies seek to expand their products into new global markets where local preferences may differ and regulations vary. However, the complexity of a single system becomes exponentially greater when there are multiple configurations of a product.

The ability to tailor products to specific customer needs make products more desirable and therefore more competitive.

The following list of software requirements will help make it easier to address this complexity, manage changes across configurations, and help your company improve the competitiveness of its product lines.

FIGURE 7: FUNCTIONAL REQUIREMENTS FOR PRODUCT LINE VARIANTS

REQUIREMENT	CONSIDERATION
Support for ISO 26550:2013 Software and Systems Engineering Reference Model for Product Line Management and Engineering	Support for the ISO standard will position a company to take advantage of industry best practices.
Automatic generation of product model	Automatic generation of models based on selected features can help support trade study analyses comparing different options and variants.

REQUIREMENT (CONTINUED)	CONSIDERATION
Component and sub-system dependencies and constraints embedded in system models	This can automate the creation of different product variants based on selected options, saving time and reducing the risk of errors.
System model drives the creation of different product variants	With the system model as a visual guideline to create different product variants, it is easier to manage complexity. Also, by using a system model, it will be simpler to visualize the impact of changes across product variants.
Variability modeling diagrams	These diagrams allow one to model different variations. The ability to use elements of variability modeling diagrams in SysML diagrams should save time and ensure consistency.
Variation options and variants are publishable to reuse library	This will aid in the creation of additional product variants and further enable reuse.

Enable Detailed Design

Forty-six percent (46%) report that communicating across departments is a top challenge of systems engineering.[4] Software can offer several functions to make it easier for teams to collaborate and work together. The next table lists the software features that will help multi-discipline teams work together more efficiently.

FIGURE 8: FUNCTIONAL REQUIREMENTS DETAILED DESIGN

REQUIREMENT	CONSIDERATION
Parallel/concurrent modeling and development	Multiple developers and engineers working simultaneously in a single database can improve efficiency. The ability to lock data prevents overwriting changes.
Collaboration tools	Inherent silos of knowledge across engineering disciplines make collaboration difficult. Technology to facilitate better collaboration can make it easier for the team to work together. Examples include automated workflows and a single source of access for requirements, test plans, design details, assets, and changes.
Management of incremental baselines	Baselines help keep all teams working with the latest information, which is essential when working concurrently.
Model comparison	As the design evolves and changes, comparing models at both the graphical and data level clarifies what has changed.
Project management tools	Project management tools are helpful to assign tasks, responsibility, and access. When working on complex projects, the team should understand who is working on what. Access control is particularly important for managing who can change what, especially when third parties are involved.
Traceability	The ability to trace the system from the high level down to the detailed design makes it easier to understand the impact of changes and implement them. It is also simpler to identify the root cause of problems found during validation. Managing and developing configurations is also easier with end-to-end traceability. Support for OSLC offers a way to link data from different systems, such as PLM, creating a digital thread across the entire system.

Support Connectivity

Even if your product is not connected or you are not currently taking advantage of IoT technologies, you should consider a solution that will also meet your future needs. McKinsey reports that based on an analysis of various forecasts, there will be a 15-20% annual growth in the number of connected objects through 2020. The McKinsey Global Institute also predicts that the impact of the Internet of Things on the global economy might be as high as $6.2 trillion by 2025.[5]

With so much growth and opportunity around IoT, you or your competitors will likely offer a connected product in the not-too-distant future. Naturally, this will create new requirements for a solution to support developing connectivity.

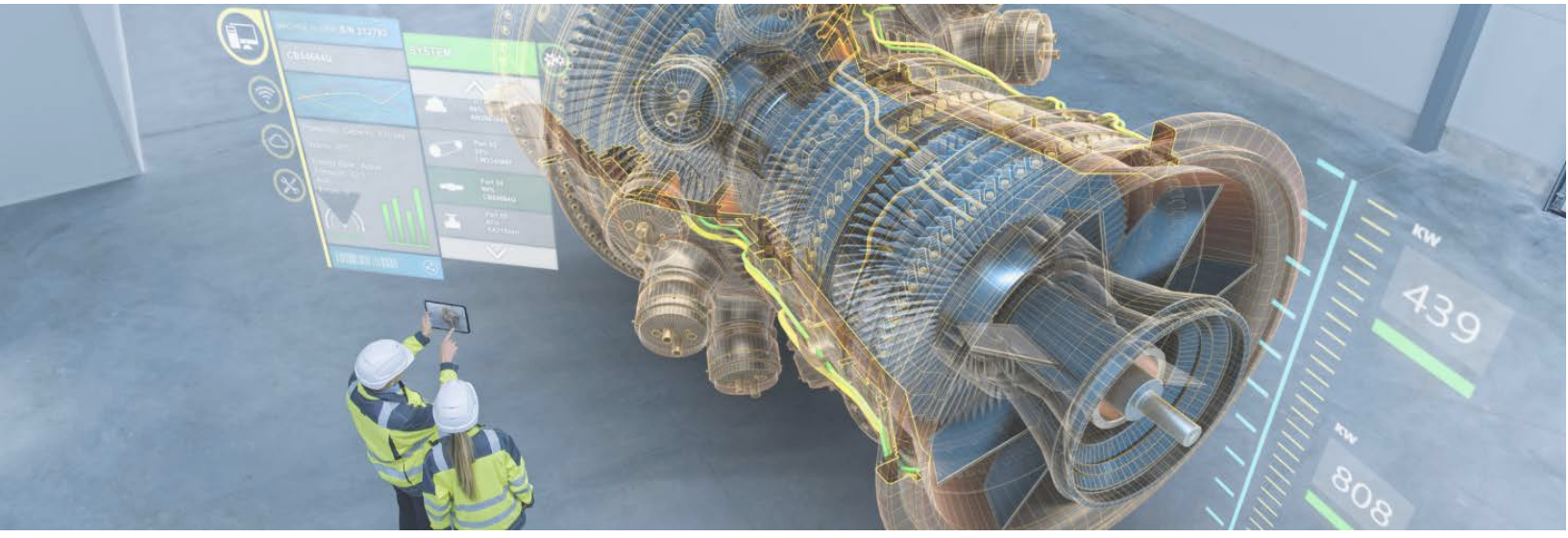
The following is a list of software requirements to support smart connected device development.

McKinsey estimates that IoT will have a $6.2 trillion impact on the economy by 2025.[5]

FIGURE 9: FUNCTIONAL REQUIREMENTS FOR SMART CONNECT PRODUCTS

REQUIREMENT	CONSIDERATION
IoT process workflow	Regulatory authorities require documented processes. The platform should have the ability to create, edit, and manage processes related to connected devices.
Embedded analytics	The big data resulting from streaming devices can quickly become overwhelming. With embedded analytics, it is easier to make sense of the data, identify trends, and recognize issues that require attention.
Code automatically generated based on defined logic	This will save time, improve quality, and minimize risk. Engineers can then focus their time on innovative functions rather than the more tedious task of writing code.
Easy to use tool for non-programmers to define device connectivity logic	Meeting customer needs with a connected device requires the input of a cross-functional team with varied expertise. Ideally, the entire team should be able to contribute and provide input into connectivity. The ecosystem around the connected device can be complicated so the tool should make it as easy as possible to take advantage of the connected technology.

REQUIREMENT (CONTINUED)	CONSIDERATION
Support for data mashups	The value of connected devices is in connecting multiple different systems. Connecting these systems and then putting key information into dashboards will make it easier to monitor devices and identify when actions are required.
Capabilities to plan the device at the system level, including defining the system architecture for hardware and software	Consider a method such as system modeling, which provides a visual method for defining devices and systems of systems. This will allow you to visually trace requirements down to the product capability that will fulfill it. Consider a systems engineering industry standard such as SysML, which is a modeling language for systems engineering.
Out-of-the-box functions to manage device performance	Using out-of-the-box functions reduces the risk of errors, which improves quality. It should include monitoring and updating devices to identify problems and fix them without disrupting product use.
Support for the latest security protocols	Keeping devices secure helps protect them from hacking.
Device sensor management	Sensors are critical for a connected device to understand its environment, making sensor management a requirement.



Verify and Validate the System

Validation involves ensuring you build the right product. Verification confirms you built the right product. With complex systems, ensuring the final product functions as originally intended is not easy. In fact, 48% report that closing the loop between requirements to validation is a top challenge.[6]

Finding problems as early as possible is an important piece of making sure the product is right. Validation should be a continuous process throughout the development cycle. This reduces the risk of finding problems at the end of the development cycle when they are more expensive to fix and can jeopardize product launch dates. In addition, regular testing will mean higher quality products that will be more likely to succeed in the market.

Functional requirements that will support early and regular validation are listed below. These functions will enable continuous validation that the product is right throughout the development lifecycle.

48% report that closing the loop between requirements to validation is a top challenge.

> Regular testing will mean higher quality products that will be more likely to succeed in the market.

FIGURE 10: FUNCTIONAL REQUIREMENTS FOR VALIDATION AND VERIFICATION

REQUIREMENT	CONSIDERATION
Test planning, definition, and authoring integrated with requirements definition	Requirement definitions are more likely to be testable when linked to a test plan. Associating each requirement with a test also leads to better test coverage, making it easier to validate that requirements are met. The final product is then more likely to meet customer and market needs.
Traceability between test cases and requirements	Linking each requirement to a test supports better test coverage as requirements without tests can be flagged. In addition, when a test fails, it will be easier to understand the impact on the requirements and the system.
Test execution software available to developers	When developers test as they code, they can catch problems earlier, leading to higher quality.

REQUIREMENT (CONTINUED)	CONSIDERATION
Tests executed in context of individual system configurations	This supports product line engineering by providing tests not just for the main platform, but also for the product variants.
Central repository for automatically capturing test results	Central repositories provide stakeholders with real-time visibility into test coverage, findings, and defects, giving them insight into issues impacting time to market, quality, and cost targets.

Assess Service Requirements

The software implementation must be successful to achieve a return on investment. Even more important, users must adopt the software feel comfortable using it. To achieve that comfort level, users should have good training resources and support when they have questions or run into problems. Finally, given the number of engineering disciplines involved in system development, systems engineering tools should be integrated with other solutions. Integration will provide a more seamless workflow across engineering disciplines to overcome the natural knowledge silos within each engineering discipline.

Integration will provide a more seamless workflow across engineering disciplines to overcome the natural knowledge silos within each engineering discipline.

The following list provides support requirements to ensure users can take advantage of the software investment:

FIGURE 11: SERVICE REQUIREMENTS

REQUIREMENT	CONSIDERATION
Qualified trainers	To ensure adoption, use trainers who are familiar with the tool(s) and standards.

REQUIREMENT (CONTINUED)	CONSIDERATION
Dedicated support staff	When help is needed, users should have access to experienced systems engineers to assist. Ensure that support hours and availability align to when users need them.
Documentation of tool installation, administration, and operation	In addition to phone and consulting support, documentation should also be an available resource.
Application programming interface (API) for integrating with other tools	Systems engineering tools cross many functions, so they should integrate with existing tools

Consider Vendor Attributes

When investing in something as strategic as systems engineering, the software vendor should be a trusted partner.

The following list provides criteria for what to look for in a vendor.

When investing in something as strategic as systems engineering, the software vendor should be a trusted partner.

FIGURE 12: VENDOR REQUIREMENTS

REQUIREMENT	CONSIDERATION
Financial stability	Is the vendor profitable? Are they making investments in R&D?
Invests in products	Is the vendor continuously investing in its portfolio to improve and expand it?

REQUIREMENT (CONTINUED)	CONSIDERATION
Support for industry standards	Given the number of tools involved in developing systems, the ability to share information across those tools will help coordinate the development process. Support for industry standards makes it easier to integrate those tools.
Has broader solutions available	Systems engineering is very broad, so tools must integrate with other engineering disciplines. Does the vendor offer extended product development solutions such as PLM and/or ALM that will scale to your needs, or are there options to integrate with another vendor's solutions?
Willing to partner with other vendors	Because systems engineering crosses multiple engineering domains, tools should integrate with other competing / complementary tools.
Flexible licensing	Ensure the licensing model meets your company's needs. Licensing schemes such as floating licenses that are released when a user exits the tool gives users flexible access. Perpetual licensing is the traditional approach, while SaaS and subscription offer flexibility to use what you want, when you need it, with less up-front capital.
Deployment options	Some companies prefer the traditional on-premise approach, while others want the benefits of a more modern cloud-based approach. Cloud-based tends to support easier collaboration, which can be particularly useful for systems engineering.
Experience	What kind of experience does the vendor have solving systems engineering problems? Do they have expertise in your industry?

Identify Specific Needs for Your Company

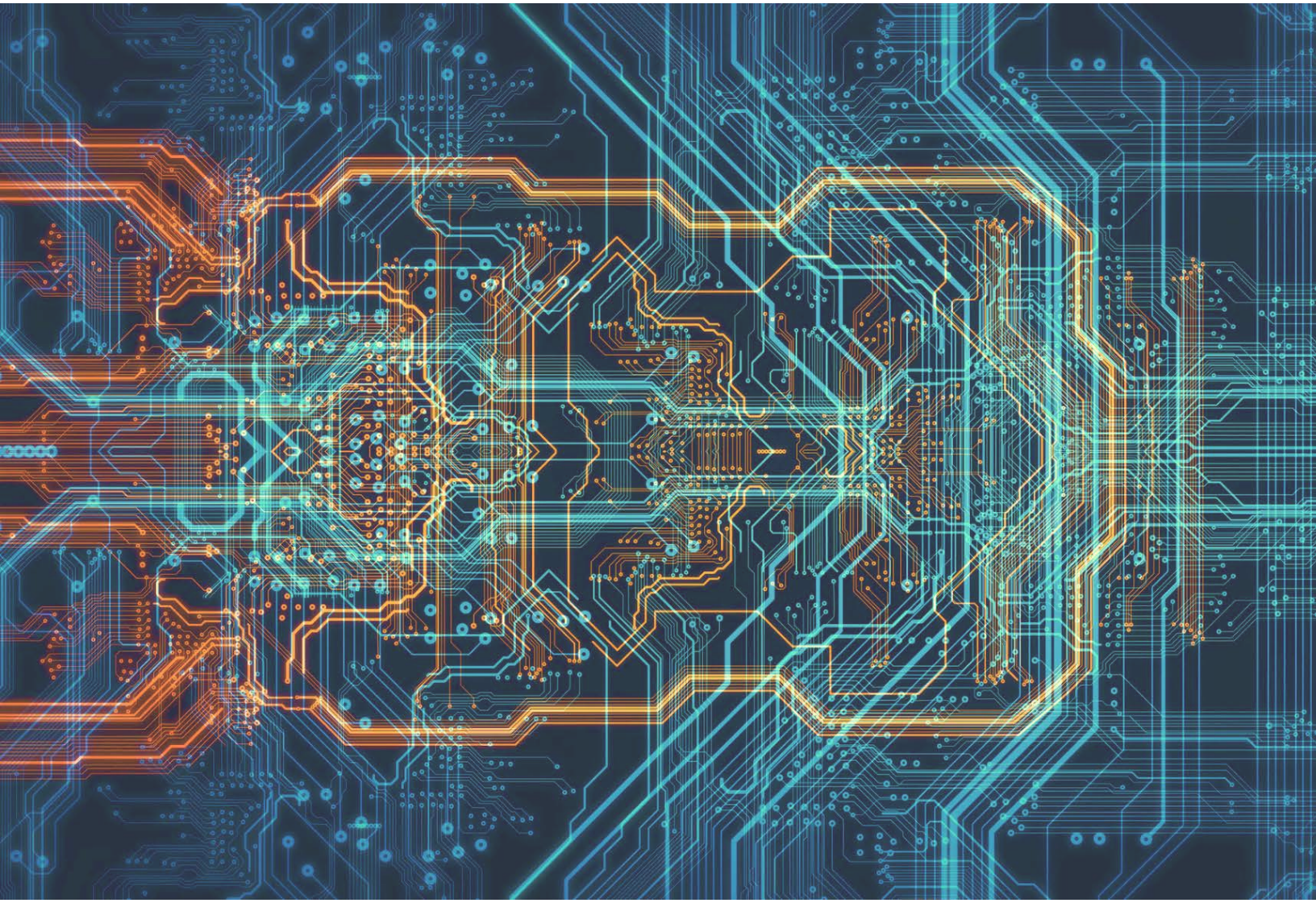
Systems engineering is complex, with many different functions involved. In some cases, your company may have existing point solutions that are working very well. For this situation, any new solution should integrate with what you have already. Also, your company or industry may have specific regulations with which you must comply. Consider other company-specific needs such as, what does your environment look like? Which programming languages do you need? Which CAD tools are used? Which PLM tools do you need to integrate with?

The following list provides a reference to think through the special considerations your company will need.

FIGURE 13: REQUIREMENTS FOR SPECIFIC COMPANY NEEDS

REQUIREMENT	CONSIDERATION
Existing enterprise tools	Will you need integrations with PLM, ALM, MBSE, etc?
Existing design tools	What solutions do you use for CAD, EDA, Simulation, and IDE?
Existing software infrastructure	Consider which existing tools must be integrated such as configuration management, data management, requirements management, engineering performance and analysis tools, other modeling tools, etc.
Support for your processes	When out-of-the-box behavior does not support your process, how easy is it to customize the solution?
Localization	Central repositories provide stakeholders with real-time visibility into test coverage, findings, and defects, giving them insight into issues impacting time to market, quality, and cost targets.
Software languages	Will you need support for C, C++, C#, Ada, Java, VB, etc?

REQUIREMENT (CONTINUED)	CONSIDERATION
Scalability	What size projects must it support, and can it scale across projects of different sizes?
Regulatory compliance	Which regulations must you comply with: ISO 26262, DO-178B, DO-178C, DO-254, SPICE, IEC 61508, IEC 62305, AUTOSAR, etc.



Conclusion

Expert systems engineering practices are vital to take advantage of innovation available through embedded software and the Internet of Things. The opportunities to create smart, connected devices can help companies set their products apart from the competition, helping them win new customers and increase revenues. However, bringing together systems of mechanical components, electronics, and software is complex. That complexity grows exponentially as companies try to meet the various needs of customers with different configurations. Connected systems add even further complexity as you add sensors, streaming data, and connected ecosystems.

Complexity increases the risk that things will go wrong. The impact of these problems can have a significant business impact and hurt product profitability. Implementing expert systems engineering practices, with the right software tools to support them, can manage this complexity, making it easier to successfully bring profitable products to market. Even if today's smart, connected devices are relatively simple, as they evolve and offer critical services such as those that impact safety, they will increase in complexity and need the same level of expert systems engineering practices.

These practices and the supporting solution are not just limited to systems engineers either. There are a variety of IoT-related roles involved with planning, designing, and architecting connected systems, such as IoT solution architects, who will struggle with the exact same challenges as systems engineers. Companies planning for growth should consider both current and future needs.

Expert systems engineering practices are vital to take advantage of innovation available through embedded software and the Internet of Things.

However, there are so many aspects of systems engineering; determining the right solution for your company can be very difficult. Using a high-level list of tool and process evaluation criteria such as the ones in this guide can help narrow down potential solutions. The lists provide a quick "litmus test" to determine if a solution and partner are a good fit before conducting detailed functional or technical reviews. In the end, it is crucial to ensure that functionality, service, vendor, and special requirements are all considered when selecting a solution.

Companies planning for growth should consider both current and future needs.

Recommendations

Based on industry experience and research for this report, Tech-Clarity offers the following recommendations:

- Identify and weigh systems engineering requirements based on company needs, existing applications, industry, and unique product and process requirements.
- Use high-level requirements such as those in this guide to evaluate solutions based on business fit before engaging in detailed evaluations.
- Consider long-term business and process growth needs and the potential to scale across product lines, departments, and engineering silos.
- Consider Systems Engineering as an upstream extension of product innovation platforms that will support a digital thread. Evaluate the potential benefits of out-of-the-box integrations either through a single vendor or options such as OSLC.
- Consider all stages of systems engineering from process, requirements, design, and validation when investing in systems engineering solutions.
- Think about all roles that can benefit from systems engineering best practices, such as IoT solution architects.
- Select a vendor who will be a trusted partner.

About the Author

Michelle Boucher is the Vice President of Research for Engineering Software for research firm Tech-Clarity. Michelle has spent over 20 years in various roles in engineering, marketing, management, and as an analyst. She has broad experience with topics such as product design, simulation, systems engineering, mechatronics, embedded systems, PCB design, improving product performance, process improvement, z mass customization. She graduated magna cum laude with an MBA from Babson College and earned a BS in Mechanical Engineering, with distinction, from Worcester Polytechnic Institute.

Michelle began her career holding various roles as a mechanical engineer at Pratt & Whitney and KONA (now Synventive Molding Solutions). She then spent over ten years at PTC, a leading MCAD and PLM solution provider. While at PTC, she developed a deep understanding of end-user needs through roles in technical support, management, and product marketing. She worked in technical marketing at Moldflow Corporation (acquired by Autodesk), the market leader in injection molding simulation. Here she was instrumental in developing product positioning and go-to-market messages. Michelle then joined Aberdeen Group and covered product innovation, product development, and engineering processes, eventually running the Product Innovation and Engineering practice.

Michelle is an experienced researcher and author. She has benchmarked over 7000 product development professionals and published over 90 reports on product development best practices. She focuses on helping companies manage the complexity of today's products, markets, design environments, and value chains to achieve higher profitability.

Michelle can be reached at michelle.boucher@tech-clarity.com. You can read additional research, watch Tech-Clarity TV, or join the Clarity on PLM blog at www.tech-clarity.com. You can also follow Michelle on Twitter at @MichBoucher, or find Tech-Clarity on LinkedIn and Facebook as TechClarity.Inc.

Michelle Boucher
Vice President
Tech-Clarity, Inc.

Acknowledgments

Tech-Clarity is an independent research firm focused on how manufacturers use digitalization, software technology, best practices, and IT services to drive operational improvement and business value. Tech-Clarity shares this knowledge with companies through publications, speaking, and strategic workshops to help company leaders understand and achieve the business value of product innovation, product development, engineering, manufacturing, service, Internet of Things (IoT), and other related software. The firm is dedicated to educating companies on making strategic improvements through the intelligent use of enterprise and digital software.

Endnotes

1. Jim Brown, "Developing Software-Intensive Products: Addressing the Innovation Complexity Conundrum," Tech-Clarity, 2012.
2. Ibid.
3. Ibid.
4. Ibid.
5. Harald Bauer, Mark Patel, and Jan Veira, "The Internet of Things: Sizing up the opportunity," *McKinsey & Company*, December 2014 http://www.mckinsey.com/industries/semiconductors/our-insights/the-internet-of-things-sizing-up-the-opportunity
6. Jim Brown, "Developing Software-Intensive Products: Addressing the Innovation Complexity Conundrum," Tech-Clarity, 2012.