IBM Watson
White Paper

Continuous engineering for the Internet of Things

Adapt time-tested methods to meet new engineering challenges

Watson IoT™

IBM

Table of contents

Executive summary

The Internet of Things (IoT) has made the leap from much-anticipated concept to exciting reality. More and more, people simply expect things to be connected, forever changing the way we interact with the physical world. As a result, businesses can operate more smoothly and profitably—and even reinvent themselves. The connected things around us can make our lives and our work simpler, safer and more convenient—and the IoT can help us tackle big problems like environmental sustainability and chronic disease management.

However, the increasing complexity of connected systems—and their inherent susceptibility to constant change—simultaneously introduces new modes of failure and makes failure more costly. This new reality means new challenges for engineering and manufacturing businesses. At the same time, the complex development environment and the rapidly evolving demands of the emerging IoT space mean that development teams must be able to continuously learn and quickly adapt.

The diversity of IoT solutions is enormous, from new products that exploit the novelty of connectivity to systems that are critical to the functioning of infrastructure such as transportation or energy systems. Similarly, the criticality of connectivity can vary enormously; connectivity can be anything from a nice feature to an essential, even lifesaving capability. Overall, as the IoT plays an increasing role in the world, reliability and security become more important. Although there is no one-size-fits-all development solution for the full range of IoT systems, robust engineering practices are vital in practically every case. This is where systems engineering can play an important role.

Systems engineering is a discipline that emphasizes coordinating and managing the complexity and dependencies of systems, subsystems and components to ensure that the desired functionality is provided and the system is dependable. Long a mainstay of the aerospace industry, systems engineering and systems thinking can be adapted to offer major benefits for the complex and fast-moving world of IoT solutions.

The core tenet of a systems approach is to coordinate and integrate codependent subsystems. This principle can extend to the development of both embedded and application software for IoT systems. IoT connectivity is increasingly a key differentiator for new products. The software that enables this connectivity is characterized by multiple environments, significant complexity and rapid change, making development a significant challenge. To address this challenge, a systems approach can be supported by the strengths of agile methods and continuous engineering—their combined abilities can result in better responsiveness, better quality control, higher productivity and shorter delivery times.

Effective IoT solution development calls for processes that are simultaneously robust, flexible and adaptable. Continuous engineering and agile principles provide a large part of the answer, and can be enhanced further by agile frameworks such as SAFe and engineering solutions from IBM. For all these reasons, the systems engineering approach is well-suited for use in software-intensive IoT system development environments.

Introduction

The hype around the Internet of Things is now rapidly giving way to the reality of implemented products and services.

Analyst firm IDC predicts that the worldwide IoT market spend will grow from approximately USD 690 billion in 2015 to USD 1.46 trillion in 2020 with a compound annual growth rate of 16.1 percent. The installed base of IoT endpoints will grow from 12.1 billion in 2015, exceeding 30 billion in 2020.[1]

Connectivity has moved from being an interesting feature to being a so-called “price of entry” requirement to achieve competitive product value and differentiation in many of today’s markets.

IoT products and services can range from the basic to the critical: cost-critical, availability-critical, brand-critical, even safety-critical. Therefore, the makers of products and services must understand and respond appropriately to the challenges of engineering for the IoT. As connectivity increases the capabilities of IoT products and services, so it also increases their complexity. New capabilities bring new failure modes. Added complexity—unless managed appropriately—can increase the likelihood of failures occurring. Furthermore, the consequences of failure can themselves be hard to predict.[2]

Therefore, increasingly critical products and services require robust IoT engineering. The primary challenges include:

- Delivering compelling functionality (where the requirements might be continuously changing)
- Delivering appropriate dependability, in the form of safety (freedom from harm), reliability (availability of services) and security (freedom from intrusion, interference or theft)
- Delivering the solution in an open context—where some of the technologies and components that contribute to the solution are not under direct commercial or engineering control
- Delivering the solution with appropriate speed and at appropriate cost to respond to competitive threats and changing market demands

IoT-related products and applications will require a more systems-oriented approach to engineering. Systems thinking, especially the concept of emergent behavior (both wanted and unwanted) is crucial for high-quality IoT development and design. Systems engineering, especially system-of-systems engineering, can help reinforce the agility and quality of IoT development and design, especially if the product being designed needs to respond to other products and systems that are not under the designers’ control.

However, systems engineering approaches must be right-sized to apply to IoT, between the two extremes of, on the one hand, extremely agile ad hoc development projects and, on the other, meticulous and expensive aerospace-grade systems engineering. Special attention must be given to safety and security aspects of IoT systems, more so than for conventional apps and software products. Tools supporting such engineering approaches must be flexible and integrated so they can provide the right amount of control and rigor, but also meet the needs of fast development cycles and time-to-market pressures.

To make the most beneficial impact on IoT development, systems engineering approaches should be part of a comprehensive continuous engineering methodology. Continuous engineering makes use of the feedback available from connected products and systems to continuously inform product refinement and new design. It consists of proven principles and practices combining systems thinking and systems engineering, embedded software development and IoT application software development, together with appropriate automation to enact those practices efficiently in a real product development environment. This paper will describe the challenges faced by teams developing IoT solutions and how these challenges can be addressed by an agile continuous engineering approach.

The characteristics and challenges of IoT products and services

Much of the power of IoT comes from the direct connection it enables between customers and businesses. This connection can transform business models by facilitating entirely new business processes or make processes much more efficient compared to traditional non-digital business processes. However, there is more at stake, because the customer is directly exposed to the effectiveness of those digital processes.

In addition to criticality characteristics, IoT applications in specific industries have different dynamics that dictate the required speed of development response. For example, small consumer products and devices such as smartphones and wearable tech might require very rapid development cycles driven by market competition and social trends. In contrast, the connected components of smart infrastructure such as a public transportation system are subject to much more strategic acquisition processes, requiring highly robust but less rapid development.

Because the IoT is a relatively new and rapidly changing area of technology, the nature of IoT development is still in its formative stages. However, there are several distinct types of IoT challenges:

- Some IoT products and services will be born on the IoT. In other words, their capabilities are only possible because of the IoT and, for companies producing such products and services, their brand will be critically dependent on the IoT.
- Some product and service concepts may change in capabilities and scope to become more critical over time, requiring that development processes also evolve to become more robust.
- Changes in regulations and standards may occur as product and service concepts become more established, leading to changing engineering and development needs.
- Many parties are involved in the development of a connected product—often an entire ecosystem of software and hardware technology component and data suppliers. These parties may be mutually dependent for success, but nonetheless work at different cadences. These extra dependencies require flexibility in development processes to assess and to respond to changes effectively.
- The “things” in the IoT must interact with other connected “things” in an ecosystem. Machine-to-machine communication and transactions will become more common and important over time.
- Public perception is always changing, and may be shifting from outright optimism in the IoT and cognitive systems to skepticism about the safety and security of such systems and concern over the wisdom of aggressive adoption.

The deep connection with the customer that is possible with the IoT not only helps foster customer engagement, but also changes product development and delivery due to:

- *Rapid feedback*
  The possibility to gain near-real-time understanding of product and service performance and customer behavior. This feedback gives businesses the insight to stay ahead of their competitors by continuously adapting to customer needs and business opportunities.
- *Rapid delivery*
  The possibility to frequently provide new and updated capabilities to customers and continually optimize competitive advantages.

The many new use cases and variables that the IoT introduces to product and service development mean that it is hard to predict exactly which development capabilities will be required over time. So a robust IoT development capability must have the means to evolve as development needs change.

Continuous engineering

Engineered systems that connect to the real world, unlike pure software applications, carry increased risk to people, property and the environment in the event of a malfunction. This risk may be compounded for IoT systems since, at least potentially, any IoT product can affect any other connected product in the world! Too rigorous an approach in a context that requires rapid product updates but with a low cost of product malfunction could unnecessarily delay product delivery, surrender a competitive advantage or result in commercial failure. On the other hand, a lack of rigor in a safety-critical or other high-cost-of-failure environment could be even more damaging. And a rapidly changing product concept—where the market size, cost of failure and future regulatory regime are unknowns—requires an approach that can efficiently adapt to needs as they change.

A vital aspect of development that is required for successful IoT development is systems thinking – an awareness of, and engineering and development response to, the demands of creating a distributed system of systems. Depending on the criticality of the product or service, this may range from collaborating more effectively between the different engineering teams and organizations involved, to much more formal and rigorous systems engineering methods.

In addition, software development is key to IoT implementation, whether this is development of embedded software for the system endpoints ("things"), or development of the overall IoT application software.

All of these engineering activities require the right degree of collaboration, traceability, automation, responsiveness to change and rigor to match the criticality of the product or service under development.

Checklist: How critical is your IoT solution?

- Is your IoT product changing your business from a traditional mechanical or electrical product manufacturer to a software-intensive product manufacturer? Do you have the in-house skills and knowledge to effect the transformation?
- Can your connected product or service directly cause harm to people or property? Examples of this would be a connected home security or heating system, or an autonomous vehicle piloting capability.
- Can your connected product or service indirectly cause harm? An example of this would be something like a connected power utility, where malfunction could damage equipment, cut off power to homes and impair safety within a population.
- Is the user experience entirely dependent upon the connected aspect of your product or service? For example, a connected taxi-hailing service cannot function without connectivity, whereas a connected car will still be functional as a vehicle without connectivity.
- Will your IoT product or service depend upon hardware and software components that are outside of your direct control with respect to engineering, commercialization or both?
- Is its availability critical? For example, a power utility smart metering system would cause moderate disruption to users if energy usage were not monitored correctly, but such a failure might be catastrophic for the grid if power demand information were not available to the utility company.
- Is the number of users very large or likely to become very large?
- Is it financially critical? Very often this may be related to the number of users, safety-criticality or availability needs of the product or service, or size of financial damages if there is a failure.

Continuous engineering uses proven practices for systems thinking and systems engineering, embedded software development and IoT application software development, together with appropriate automation to enact those practices efficiently in a real product development environment.

Continuous engineering isn't an all-or-nothing approach; it is intended to be scalable to the needs of IoT product and service development, and modifiable in numerous dimensions:

- Across different team sizes and development ecosystem structures from the simple to the complex
- Across different development disciplines of systems engineering, embedded software development and IoT application development
- Across different engineering activities from requirements elicitation and management to system integration and testing

By prioritizing the needs of a project, continuous engineering can be applied where it will deliver the maximum return and its use can be modified as development needs and priorities change.

Systems engineering for the IoT

Discover the benefits continuous engineering can bring to systems engineering for the IoT. Click here for a video.

Systems engineering is a vital aspect of an overall continuous engineering methodology. IoT systems engineering is all about coordinating and managing the complexity and dependencies of IoT systems to ensure that the desired functionality is provided and the system is dependable. Systems engineers are responsible for achieving two main goals:

- All system requirements are met, including user, market and regulatory requirements.
- The system is delivered in a timely and cost-effective way to meet the business's technical and commercial objectives.

Systems engineering must also ensure the system continues to meet these objectives in a dynamic deployment context. Systems engineering has been practiced since the early days of the American space program and was instrumental in its success. It has since grown into a discipline that is active in many industries. Systems engineering for the IoT must be exceptionally agile, but without losing the key characteristics that make it valuable.

Systems engineers know that to be successful, a connected system must be engineered not only to work correctly, but also to meet key qualitative goals, such as system reliability, resiliency, safety, security and maintainability. Each of these so-called non-functional requirement areas calls for its own kind of engineering practices, tools and verification approaches. Furthermore, verification and validation must be adapted to the challenges of IoT systems of systems. Implementing a system such as a smart thermostat may appear simple, because it seems like the thermostat is only responsible for controlling heating and cooling for the building in which it is installed. However, when the thermostat is connected via the IoT, it requires a comprehensive systems engineering approach to ensure that occupant safety is not compromised and the buildings in which the system is installed don't risk serious damage from hackers, communication issues or errant software updates.

Systems engineering holds many of the solutions to IoT engineering challenges, so long as systems engineering itself can evolve. Historical systems engineering processes, which were intended for long, careful aerospace development programs, are not a good match for dynamic IoT systems. Development teams may be distributed across locations, time zones or countries. Markets may be unforgiving when developments take too long, and customers won't long tolerate difficult or problematic installation and operation. Systems engineering must be right-sized for the risk associated with a particular IoT application and systems engineers must employ agile, lean techniques which bring the needed rigor without aerospace-sized overhead.

Certain capabilities are applicable to systems engineering in general, as well as to embedded software development and development of IoT applications in particular:

- *Collaboration, workflow planning and management*
  Ensuring that systems engineers and other engineering disciplines and stakeholders are on the same page throughout the project and work together effectively as changes occur without duplication of effort or omission of needed activities
- *Requirements management and traceability*
  Ensuring that stakeholders' needs are correctly captured, understood, recorded and used to drive the development of appropriate solutions. Traceability must exist not only between systems engineering and design but also between downstream test and integration.
- *Change and configuration management*
  Supporting different product versions and the different product configurations that differentiate the members of a product family
- *Model-based systems engineering*
  Ensuring that potential solutions are understood and evaluated early and that effective trade studies are conducted to select an appropriate solution
- *Systems quality management*
  Ensuring that verification and validation activities are derived from system requirements and are conducted early and often as changes occur
- *Lifecycle information management*
  Using detailed traceability between engineering artifacts to improve engineering outcomes through better decision-making
- *Agile planning and management*
  Supporting the ability to efficiently decompose development into discrete tasks, dynamically prioritize those tasks, and apply iterative development to deliver tested and working components
- *Release and deployment management*
  Balancing the need for speed in delivery of updates with the capabilities of the IoT infrastructure and the need for dependability and control of software deployment
- *Continuous feedback*
  Using information sources, such as operational performance, error and defect information, and user behavioral available from connected products and systems to continuously inform product refinement and design evolution

Embedded software design for the IoT

Discover the benefits continuous engineering can bring to embedded software development for the IoT.
Click here for a video.

Embedded software development is the creation of software for the physical devices that make up components of an IoT system. Devices can range from small-scale sensors to complex physical products such as airplanes and medical equipment. Embedded software is increasingly responsible for the value and differentiation of such products. This is no different when those products are IoT endpoints, but the IoT context intensifies many of the challenges facing developers of embedded software.

IoT components typically have more dependencies than other domains of software development:

- The operating environment of embedded software is closely coupled to the hardware of the device, meaning it is dependent on the hardware design.
- User experience is critically dependent on embedded software.
- Embedded functionality is augmented with off-board functionality in the IoT application software.

In an IoT context, embedded development becomes more dynamic:

- Embedded software updating moves from being a manufacturing or service activity to an in-service or even real-time activity.
- Operational and user data can be fed back in near real-time into software development to inform product improvement. This requires a development approach that can respond rapidly and efficiently to new information.
- Functionality can be distributed in new ways between embedded software and software running on an edge device or cloud server.

Managing dependencies, coping with the dynamic nature of development and supporting agile approaches require a mixture of continuous engineering capabilities according to the needs of the particular project.

In regulated (for example, safety-critical) environments, not only must traceability exist within development processes but it is vital to be able to use that traceability to extract development artifacts to demonstrate compliance with standards. Automation in both generating traceability links and following those links to extract compliance reporting information is vital to minimize errors and contain the cost of compliance.

IoT application software development

Discover the benefits continuous engineering can bring to IoT application software development. Click here for a video.

IoT application software development is the creation of the software that coordinates the components of an IoT system to deliver the overall system functionality. It is typically a combination of local, cloud-based and mobile components and integrates many technologies including data acquisition and storage, analytics, user and access management and security, business process automation and more. IoT applications go beyond conventional business applications and product software and are central to the business model for an IoT-based product or service. For this reason, application dependability and the ability of the development team to respond to changing demands are critical to success.

Even though an IoT solution may include complex systems of systems, the expectation is that it will respond in real time. IoT software development is distinguished from software-only development by several needs, including:

- Early development of the architecture
- A focus on critical architectural mechanisms
- Physical and virtual deployment considerations
- Functional integration with embedded software in IoT endpoints (or “things”)
- Incorporation of development analytics into the ongoing development process

IoT teams tend to be interdependent, distributed and cross-organizational. They can exist for significant periods of time.

An IoT development project is more than the sum of lots of smaller software projects. The permutations of integrations between elements of the deployed system cause a geometric increase in complexity. The number of variables such as device platforms, micro-services, languages, libraries and more, adds multipliers to the already complex process of developing software. The result is that navigating across artifacts, tools and schedules requires more intelligence and support.

Security, privacy, scalability, deployment and integration are all areas that must be considered, documented and clearly solved in an IoT system. In addition, these aspects must be reassessed when new elements are added to the system though product introduction, evolution or ecosystem expansion.

Continuous engineering and agile methods

Continuous engineering isn't a single process; it's a mindset that guides the application of detailed design and development processes. Therefore, continuous engineering can be implemented according to agile methods, and this combination is well-suited to development for the IoT.

Agile methods are well-established for many areas of software development, delivering consistent high productivity, high quality and attention to design while also enabling responsiveness to change. These characteristics are, if anything, more important for IoT software development, making agile methods attractive to companies developing for the IoT.

In recent years there has been recognition that agile methods apply not only to software, but increasingly to systems engineering activities.

Agile methods offer a number of benefits:

- By prioritizing collaboration with clients, agile can help ensure that the right solution is delivered.
- By dynamically managing development priorities, agile can help maintain focus on the project outcome when there is a large degree of uncertainty regarding the direction of the correct solution in a project. This is a significant possibility in IoT development.
- By avoiding technical debt and delivering only tested and working products, agile can avoid systems integration issues getting out of hand, which could lead to project failure.
- By eliminating unnecessary work, agile can help ensure projects are delivered efficiently.

Agile techniques can be adapted for use in IoT development, but these must be selected and applied with care. Some agile techniques were originally designed for small, independent, co-located teams working on projects that were presumed to have limited life. These techniques can increase the risk of failure in an IoT project. But other techniques and tools are better-suited to use on IoT projects. For example, the Scaled Agile Framework (SAFe) provides tested mechanisms for minimizing risk while applying and scaling agile methods to IoT projects. From SAFe v4.0, the framework includes support for agile systems engineering.

The IBM Internet of Things continuous engineering solution

The IBM® Internet of Things continuous engineering solution provides a common development solution for systems engineers, developers of embedded software and developers of IoT application software. Built-in data visualization, organization, analysis and reporting tools are designed to help users to make sense of and share complex engineering data.

Other capabilities from both IBM and third-party vendors can be added using the solution's Open Services for Lifecycle Collaboration (OSLC) support. The solution also includes customizable process support with preconfigured process content available for systems engineering and software teams, agile (SAFe) support and numerous industry standards.

The solution's key capabilities apply to the needs of systems engineers, embedded software developers and IoT application software developers, as shown in Table 1.

Table 1. Key IBM IoT continuous engineering solution capabilities and their applicability to three IoT engineering activities (more check marks indicate more applicability)

Capabilities			
	Systems engineering	Embedded software development	IoT application software development
Collaboration, workflow planning and management	✓✓	✓✓	✓✓
Requirements management and traceability	✓✓	✓	✓
Change and configuration management		✓✓	✓✓
Model-based systems engineering	✓✓	✓	
Systems quality management	✓✓	✓✓	✓✓
Lifecycle information management	✓✓	✓✓	✓
Agile planning and management	✓	✓✓	✓✓
Release and deployment management		✓✓	✓✓
Continuous feedback	✓✓	✓✓	✓✓

When the solution is deployed throughout your business, each engineering discipline can work on their required engineering practices using processes designed for their needs. Furthermore, because the underlying tools are common, engineers in different disciplines can more easily collaborate and share information. The result can be better-connected development.

Conclusion

The IoT offers the prize of new possibilities for product function and new value for both the users of those products and the businesses developing those products. But just as the value of IoT systems is more than the sum of the parts, so the challenge of engineering those systems is more than the sum of the component engineering activities.

The complexity and interdependencies that characterize IoT systems require a tight coupling between the engineering disciplines responsible for architecting and implementing their functionality. Furthermore, IoT solutions do not fit the conventional design/manufacture/sell business model; immediate feedback from the operational environment and accelerated delivery of design changes to deployment both demand a highly responsive engineering capability.

...just as the value of IoT systems is more than the sum of the parts, so the challenge of engineering those systems is more than the sum of the component engineering activities.

The key areas of engineering that must be connected to deliver successful IoT products are systems engineering, embedded software development and IoT application software development. Continuous engineering connects these activities through agile, collaborative processes supported by common tooling to provide automation, management of engineering artifacts, data and traceability, and analytics-driven engineering insight to support better decision-making.

Continuous engineering is not a fixed set of processes; it can be adapted to support the needs of different industry standards and for teams working at different delivery cadences, even within the same project.

Agile approaches are important for optimizing engineering delivery for the IoT. However, agile methods must be adapted to the scale and rigor of IoT development. The Scaled Agile Framework can be used to implement agile continuous engineering processes for IoT solution development.

The IBM IoT continuous engineering solution delivers the tools and agile processes to provide a scalable, frictionless environment in which systems engineers, embedded software developers and IoT application software developers can work together while balancing the need for dependability with speed and cost of delivery.

For more information

To learn more about IBM solutions for IoT engineering, please contact your IBM representative or IBM Business Partner, or visit: ibm.co/eng5

Endnotes

[1] Worldwide Internet of Things Forecast Update2016-2020, IDC www.idc.com/getdoc.jsp?containerId=US40755516

[2] Nest Thermostat Glitch Leaves Users in the Cold, NY Times, 13 Jan 2016 www.nytimes.com/2016/01/14/fashion/nest-thermostat-glitch-battery-dies-software-freeze.html?_r=0

WWW12373-USEN-00

Watson IoT™