

Agile Fundamentals  
[Page 3](#)

Beyond Software  
[Page 6](#)

The Systems Engineering  
Challenge  
[Page 9](#)


A Truly Agile Organization  
[Page 13](#)

# Beyond Software: Agile for Product Development

Page 2

# Beyond Software:

## Agile for Product Development



Agile methods have proven effective for software development and have grown in popularity across a number of software disciplines, including application and enterprise software, as well as real-time and embedded software. Agile methods build capabilities using an iterative approach, as opposed to traditional approaches where detailed requirements are defined early and designs are completed in full before a single line of code is written. Agile incrementally defines requirements with priorities, partial component designs and working systems that evolve through iterations called sprints.

The question remains, however: Do the 12 agile principles apply to more than just software? For example, can those principles be

applied to other products, such as physical components that contain both electronics and software engineering aspects, and possibly a mechanical engineering aspect as well?

Agile principles are in line with (and in many ways borrowed from) lean management's approach, which includes building a factory floor feedback loop, breaking down manufacturing into smaller components, continually adjusting for bottlenecks, and changing processes based on customer feedback and inventory levels. Given that, it seems as though agile principles should directly apply beyond software development. However, the answer is not straightforward, because differences between software and physical components affect how agile is applied. Let's explore how agile can be applied in light of these differences.

### Agile's 12 Principles

1. Customer satisfaction
2. Changing requirements
3. A working, evolving product delivered frequently
4. Cooperation among all stakeholders
5. Individual motivation
6. Face-to-face conversation
7. The product as the principal measure of progress
8. A sustainable pace of work
9. Good design
10. Simplicity
11. Self-organizing teams
12. Adaptation to change

# Agile Fundamentals



Agile’s 12 principles form the foundation for four focus areas that can be applied to product development and systems engineering projects. Although we’re interpreting these principles in the broader context, they need not change to support the agile focus areas that can be derived from them. These focus areas include:

## 1. Focus on Individuals and Interactions

Agile focuses on the individuals who do the actual work, as well as their interactions with one another, the customers and end users of the product being built, and other stakeholders. The teams formed around

product construction should be self-organized and self-motivated, meaning that walls between those who define the requirements, those who build the products, and the customers and end users should be removed where possible. Further, the reasoning for why things are done, and when they’re done, is a natural result of prioritization that everyone agrees to.

## 2. Focus on the Product

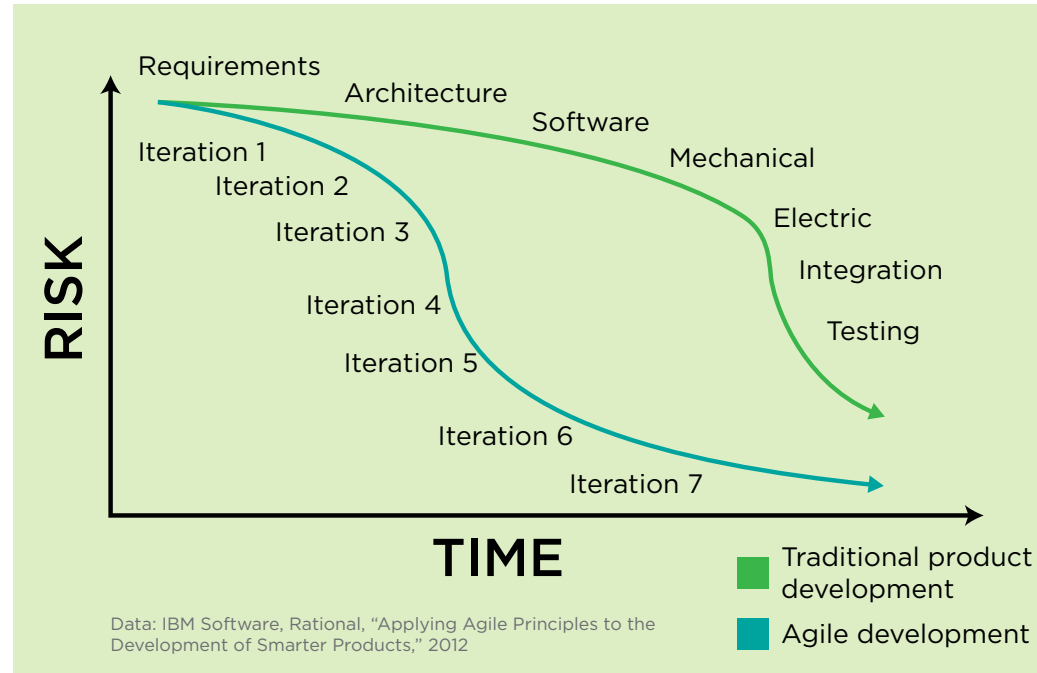
Agile’s aim is to build a working product that evolves rapidly and iteratively, which is a key factor in motivating teams to build a product

that best meets a customer need. The goal is to measure the value in product requirements, make them verifiable and deliver a product that meets those requirements. Building portions of the product iteratively using subsets of high-priority requirements creates a feedback loop where the product is rapidly refined while it is being constructed.

### 3. Focus on Customer Collaboration

An agile product development process shifts from one based on a static list of requirements with a fixed contract to one where the customer and builder collaborate continuously. Contracts may still exist, but with agile they focus on objectives to be met instead

**Figure 1: Agile vs. Traditional Development**



Agile development can prioritize iterations to reduce risk earlier than traditional product development processes.

of the solution. The result is a feeling that all stakeholders are “in this together,” with a common desire to achieve a shared vision, where everyone participates as a team and makes trade-offs during the entire product build cycle.

As Dave (Doc) Brown, chief architect for systems at IBM Software Services for Rational,

says: “It’s not about defining a detailed plan up front for the 100 or so things that need to be done. It’s more an idea to do something about all of these things, with a plan to deliver them in phases and adjust as the work is done.”

### 4. Focus on Responding to Change

Change occurs everywhere and constantly, and those who adapt to change tend to be more successful than those who fight it. The same principle holds for product development. Having the ability to identify the need for change and then rapidly respond to it is a key focus of agile. The need for change results from the knowledge gained both by the development team and the stakeholders during rapid iterations, with demonstrations, in a continuous product build cycle (i.e., a feedback loop), combined with constant customer collaboration. Agile can help organizations overcome cultural issues related to customer collaboration and can also help in situations where customers may not be

directly accessible to all team members because of their location, role in the organization or other circumstances.



“It’s not about defining a detailed plan up front for the 100 or so things that need to be done. It’s more an idea to do something about all of these things, with a plan to deliver them in phases and adjust as the work is done.”

Doc Brown, IBM Software Services for Rational

### The Intersection of Lean and Agile

Lean is a development concept that emphasizes the elimination of waste and focuses on nothing but tasks that produce demonstrable value, such as something that a customer would pay for. The seven lean principles are: eliminate waste, amplify learning, and enable just-in-time decisions, fast delivery, teamwork, systemic integrity and wholeness (i.e., consider components, their interactions and their origins as part of

the end product). Lean can be useful to some groups who aren’t ready to go agile, such as those designing and building hardware. It also

can be a way to get closer to agile by cleaning up existing processes.

Agile and lean complement each other in many ways. They focus on performing tasks that produce the most value; building teamwork and trustworthiness; and creating a learning-enabled feedback loop, faster delivery and a fail-fast mentality where defects and needed changes are identified early.

# Beyond Software

Once you begin to think outside of the software domain, you see that agile can also be applied to the production of many types of products, such as embedded hardware, appliances and other physical components in an overall system. Organizations adopt agile for many reasons, not the least of which is producing higher quality products. (For the full list of reasons, see p. 7.) For physical systems, higher quality means not only correct functionality and high reliability, but also improved longevity, repairability, usability and aesthetics, among other areas. These factors can all be improved and ensured through an iterative process where product refinements are made over time, with end-user involvement and feedback.

Amit Fisher, systems technical client relationship manager at IBM Software Group, Rational, is careful when speaking of agile in systems and physical component development, but he's positive about its impact on quality and correctness. "Many people think you can take something that's a success in application development and apply it as-is to system design or mechanical design, and you'll get agile inside the systems domain. The fundamental difference is the artifact," Fisher says. "These artifacts include documents, architectures, models, tests and



so on, and are not executable by nature (like software). This challenge needs to be overcome, and only then will a feedback loop from the requirements to the implementation be feasible.”

“Agile is a rigorous discipline, which fits well in the systems space,” Fisher says. Making artifacts verifiable and executable adds to agile’s success in electronics and mechanical engineering. With many systems projects — such as designing a car — iteratively creating and verifying a complete physical end product isn’t feasible. Instead, you need to focus on the requirements and design. “The secret sauce for agile here is technology that creates requirements and design artifacts that are executable and verifiable, in order to get feedback early,” Fisher explains. “New innovation and tools are available to help.”

This concept isn’t new; engineers have relied upon modeling, simulations and mathematical algorithms for systems architecture and design for years. Anywhere you have structured design, math can help to prove its correctness. In mechanical design specifically,

it’s entirely possible to model, test and generate user feedback iteratively on new designs well before a physical prototype is built. The same applies to CPU design and many areas of aeronautical engineering: computer-aided modeling and design allow you to create and verify systems before they exist physically. In this sense, agile has been applied in these use cases for a long time.

“Nine out of 10 customers I speak to express the need for better, more accurate design, earlier,” Fisher says. “The goal is to declare the system ‘correct by construction’ because it went through an iterative, proven design process.” It’s possible to use computational engineering to validate that the behavior of a system will be as planned. The tools to achieve this exist to a good extent, and many companies are implementing these principles.

### **Agile for Research and Development**

Research and development is traditionally viewed as a long-running process with many failures and false starts. Agile incorporates a

## **Organizations Adopt Agile To:**

- Help address integration issues early in the product life cycle
- Provide a constant feedback loop
- Refine, prove and correct requirements earlier rather than later in the process
- Help everyone deal with change
- Increase quality from many viewpoints
- Become more competitive through quicker product release cycles
- Increase productivity with gains in efficiency, not hours worked
- Adapt to unique and diverse projects
- Avoid pitfalls common to other approaches, such as integration nightmares at the end of the development cycle and discovering too late that a product doesn’t meet the customer’s needs

“fail fast” approach that’s in line with lean concepts. By taking it further and making R&D more



“Agile is a rigorous discipline, which fits

well in the systems space. ... The secret sauce for agile here is technology that creates requirements and design artifacts that are executable and verifiable, in order to get feedback early.”

Amit Fisher,  
IBM Software Group, Rational

iterative and agile, key learning — even from failures — can happen sooner and be applied to real product development before the R&D effort is deemed a success or failure.

“In theoretical R&D, difficulties can arise if there’s no stakeholder or product owner,” says IBM chief systems architect Brown. In agile, everyone has a role, including the product owner. The person in this role represents the customer when defining requirements and testing each release. “In pure research, without a marketing or business driver, the role of product owner can be difficult to fill,” says Brown. “When you

incrementally build something, who do you demonstrate it to? You need a product owner.

Without that clear and definitive guidance toward a specific target, prioritization can be more difficult, and the project can constantly change direction — never reaching a clear goal or result.”

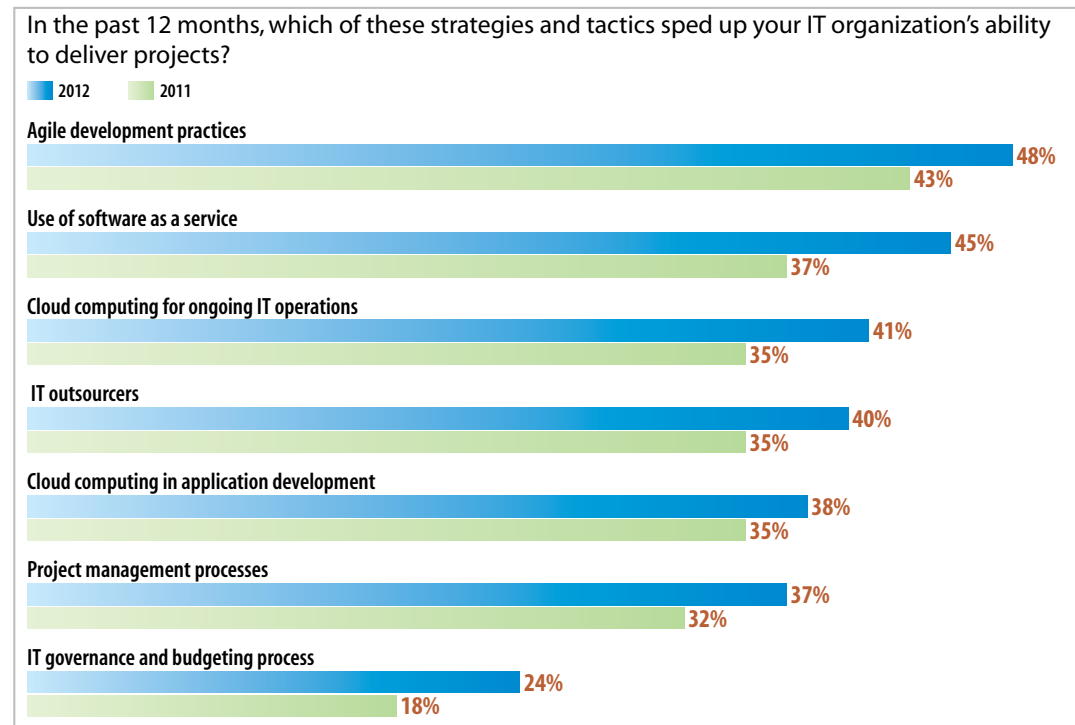
Research by nature is an agile process and has been since well before agile was a defined practice. The early exploration of new concepts

involving prototypes and revisions is agile by definition. Most researchers begin with a high-level goal in mind, but often the process leads to new discoveries and other useful artifacts. This forces them to respond quickly to changes in scope and new priorities — another key aspect of agile.

Fisher agrees that R&D shouldn’t be done in a vacuum, and customer involvement

is needed throughout. “At IBM, applied research uses the field as its lab,” he says. “You cannot truly innovate unless you have an end customer who matters.” Even with new ideas that emerge without direct customer involvement, Fisher emphasizes the need to get customers involved early to validate those ideas and help them evolve.

### Figure 2: Agile Delivers



Data: InformationWeek Outlook Survey of business technology professionals October 2011 and November 2010 adopting specified strategies and tactics





# The **Systems Engineering** Challenge

The systems engineer (SE) is responsible for gathering requirements and coordinating overall systems design, including overseeing product engineering, coordinating component design and development, and managing the end product delivery from start to finish. The SE must ensure that the requirements are correct and well communicated, and that all the engineering disciplines involved (electrical, mechanical and so on) integrate properly to meet the customers' needs.

The objective of systems engineering is to cut risk and reduce development time while delivering better quality. The SE does this by applying "systems thinking" up front to flush out architectural issues early (as in, to fail fast), and to avoid problems discovered late in the process, which has a lot in common with making sure the right product is delivered. There's a clear alignment between systems engineering and agile objectives. As businesses find they must develop more complex systems and deal with ever-changing customer market



demands, applying agile is increasingly appealing. Agile is quickly becoming fundamental to success in systems engineering thanks to its iterative feedback loop, stakeholder collaboration, continuous evaluation and risk reduction.

Modeling tools and computer-aided hardware development let you begin to integrate and test system components sooner. Agile capitalizes on the benefits of these techniques to also reduce dependence on physical prototypes. With aircraft design, for example, you're forced to create a testable, capable design before building a hardware prototype. This approach to agile, using models and simulations to iteratively integrate component designs, is a viable alternative when factors such as cost

render the iterative construction and integration of physical components infeasible.

"We can [apply] computational engineering [with computational models] in order to validate that a system will behave as planned and show it to the customer," Fisher says.

Tools to do this exist. However, there are some gaps, mainly concerning education

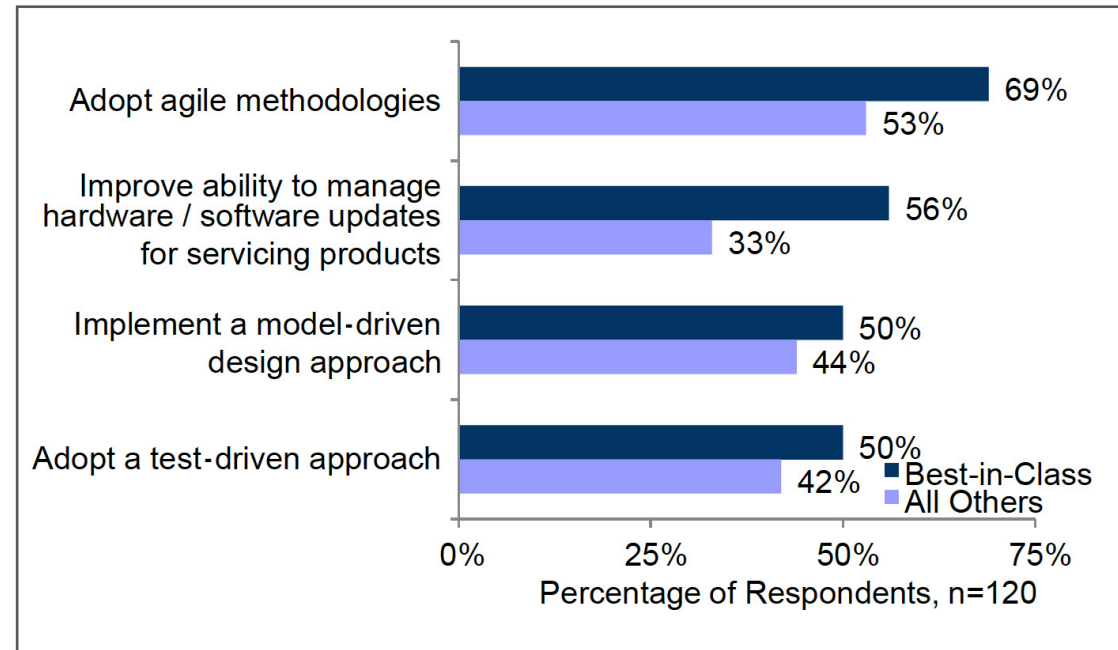
and the workforce's willingness to use these agile tools and techniques.

Let's look at some of the benefits systems engineers get from coordinating the construction and integration of complex physical system development. These benefits apply to all artifacts of systems engineering, including requirements, architecture and design, models and simulations, as well as the physical components of the system (electrical and mechanical).

### Increased Predictability

Using agile processes for the development of hardware and other physical systems can be a challenge. But agile principles can be applied around the "fixed points" of hardware development and can make the design process more responsive to unpredictable influences. For instance, you can apply agile principles to the architecture process, using modeling and simulations. Agile also suggests

**Figure 3: Strategic Actions of the Best-in-Class Performers to Improve Embedded Software Development**



Data: Aberdeen Group, "Embedded Software Development," Sept. 2012



“When you understand how to test a system or component, then you truly understand what needs to be built, and the more accurate your understanding of the requirements is.”

Doc Brown, IBM Software Services for Rational

a modular, subsystem approach to physical system prototyping and development. For

example, you can integrate electronic subcomponents and mechanical artifacts as they're delivered early during the development process, as even very large physical systems are made up of smaller parts.

Agile can be applied to systems engineering in such a way that requirements become more detailed incrementally and can be provided to engineers over

time, Brown says. Requirements should be verifiable, where possible, using formal practices such as modeling or mathematics. The resulting components can then be integrated incrementally, avoiding the scenario with other approaches where integration occurs toward the end of the project. Because you've been integrating all along, the

project's outcome is more predictable. This applies to the components of large-scale physical development as well.

In a recent joint project of IBM Research and Airbus, Fisher says, critical components of a new aircraft were modeled incrementally, including aircraft doors and flight-control surfaces, as well as the network of sensors, power supply lines and data communication buses across numerous physical locations. Add to this the need to adhere to safety and performance regulations, and the problem quickly became impossible to solve using paper or a whiteboard.

Instead, an agile-based process with mathematics, optimizations and other tools made it possible to model the problem and continually optimize and improve it until the correct solution was delivered. In the end, collaboration, prototyping and iterative machine design helped achieve the project's goals more predictably, leading to lower costs and higher quality. All the engineers involved said the agile process was key.

## Early Risk Reduction

An agile-based process lets SEs use a test-driven approach that encourages measurable development progress and promotes the early validation of requirements. Testing can be a challenge with physical systems development, but modeling and simulation can help. Chief architect Brown emphasizes the need to think about testing from the beginning, as part of high-level planning.

“When you understand how to test a system or component, then you truly understand what needs to be built, and the more accurate your understanding of the requirements is,” Brown says. You must define the tests with the requirements and validate them iteratively with the customer. “You're in effect testing your agreement over the requirements. And if you're in agreement, then you're building the right thing,” he says.

## Increased Quality

Continuous integration has a role in product and systems development for hardware subsystems,



product lines, hardware accessories, and even firmware and device drivers. It's important to begin an iterative, continuous-integration effort as early as possible to avoid the issues that occur when integration is saved until the end. Essentially, you're extending the arguments for modeling and test-driven requirements verification and development into the continuous-integration process, which continually works to improve quality.

### **Better Configuration Management**

Continuous, incremental integration stresses the need to manage all relevant aspects of engineering in a way that's tightly integrated with task and workflow management. This involves tracking changes to all artifacts, including requirements, designs, simulations, hardware and software. Configuration management tools and techniques should be at the heart of the feedback loop of a self-documenting project, providing feedback to the entire team each time a change is made.

The use of configuration management and

other agile workflow tools is fundamental to ensure that software component builds are automated, changes to all artifacts are planned and traceable, and all stakeholders are in sync in terms of overall integration progress and changes. This approach includes capturing the results of simulations, automated software builds and predefined tests executed at the time of system integration for each sprint.

### **Improved Planning Across Product Lines**

Change can have far-reaching effects when you're working on related hardware components and products. For example, when planning a line of related products, any one change can ripple throughout the line. The later a change is required, the more costly it is to implement, since it may affect a larger number of products.

Instead, an agile approach to product-line definition lets you begin the construction of new products before all of the product-line definitions are complete. The sooner the construction process begins, the sooner the

customer feedback loop can be put in place to help evolve the product line before these plans are too far along.

Brown describes the agile process that can help here: "The idea is to have sets of plans — a high-level plan for the product or product line, and a series of detailed plans that usually line up with your agile sprints." This high-level planning effort is part of what some organizations call "sprint zero," where the product or product-line requirements definition begins. However, this doesn't mean the construction process has to wait until all detailed plans are complete. Instead, detailed plans are created and executed incrementally, gathering feedback from users to refine the plan as you define it.

# A Truly Agile Organization

Applying agile to disciplines beyond software isn't an easy, one-step change. There's no clear, simple transition from legacy processes to agile. However, there is agreement that agile principles will improve these processes. As systems grow more complex, with more features and mostly flat budgets, improvements to development processes are needed. Agile can help to meet that need for continuous improvement.

By applying agile to systems engineering, you can help to more quickly and accurately refine requirements, produce tests for those requirements, prove designs with working models, increase collaboration with customers and reduce difficulties associated with fixed time and budgets. This avoids nightmare scenarios where integration happens at the end of a project, revealing major problems that can lead to project failure or, at best, derailed timing and degraded financial performance. Instead, agile helps identify changes that are needed earlier in the process, reducing risk sooner and enabling more predictable delivery.

Given that agile principles have been proven to work in software development, the time is right to consider their application to systems development. Although it may require cultural and organizational changes, agile can deliver business and technical benefits, helping to achieve continued improvements in quality, productivity and predictability in the face of ever-increasing complexity and market demands. Process improvement is a necessity, and agile delivers the proven results that benefit you, your customers and your business. ■

## [Resources]

To learn more about how IBM is defining new approaches to software development that can help you create smarter products, read the following reports:

- *Accelerating Innovation*
- *Building Smarter Products with Product Development Capabilities from IBM*
- *Applying Agile Principles to the Development of Smarter Products*