UBM Tech

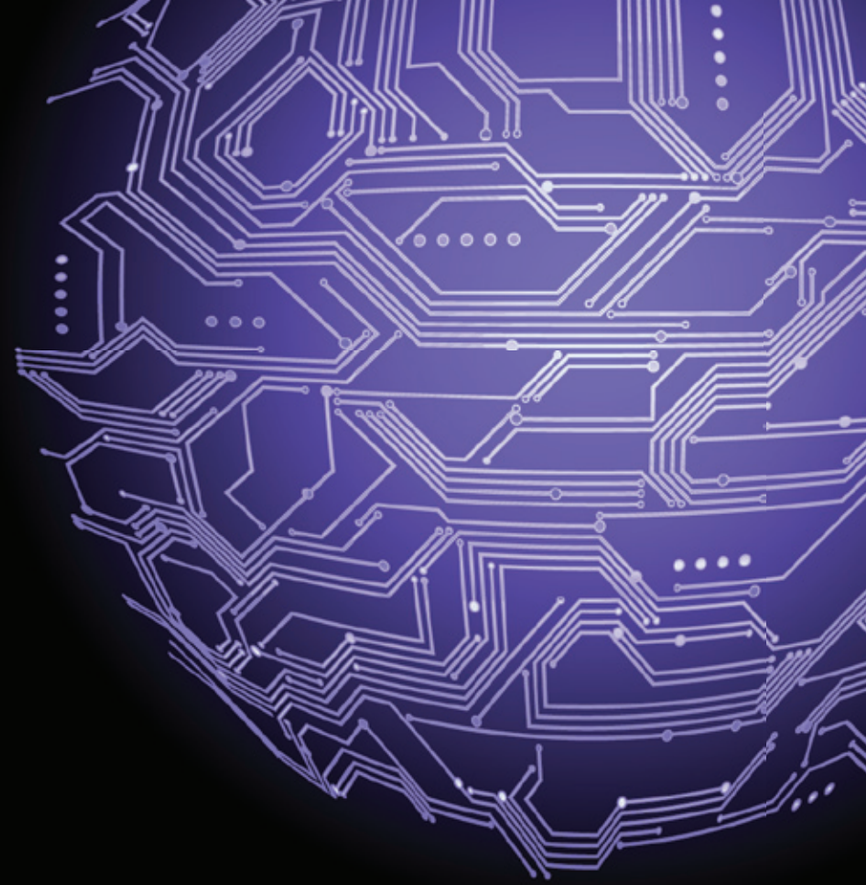
June 2013

INSIDE

Agile in the Embedded World

SPONSORED BY IBM

Agile in the Embedded World

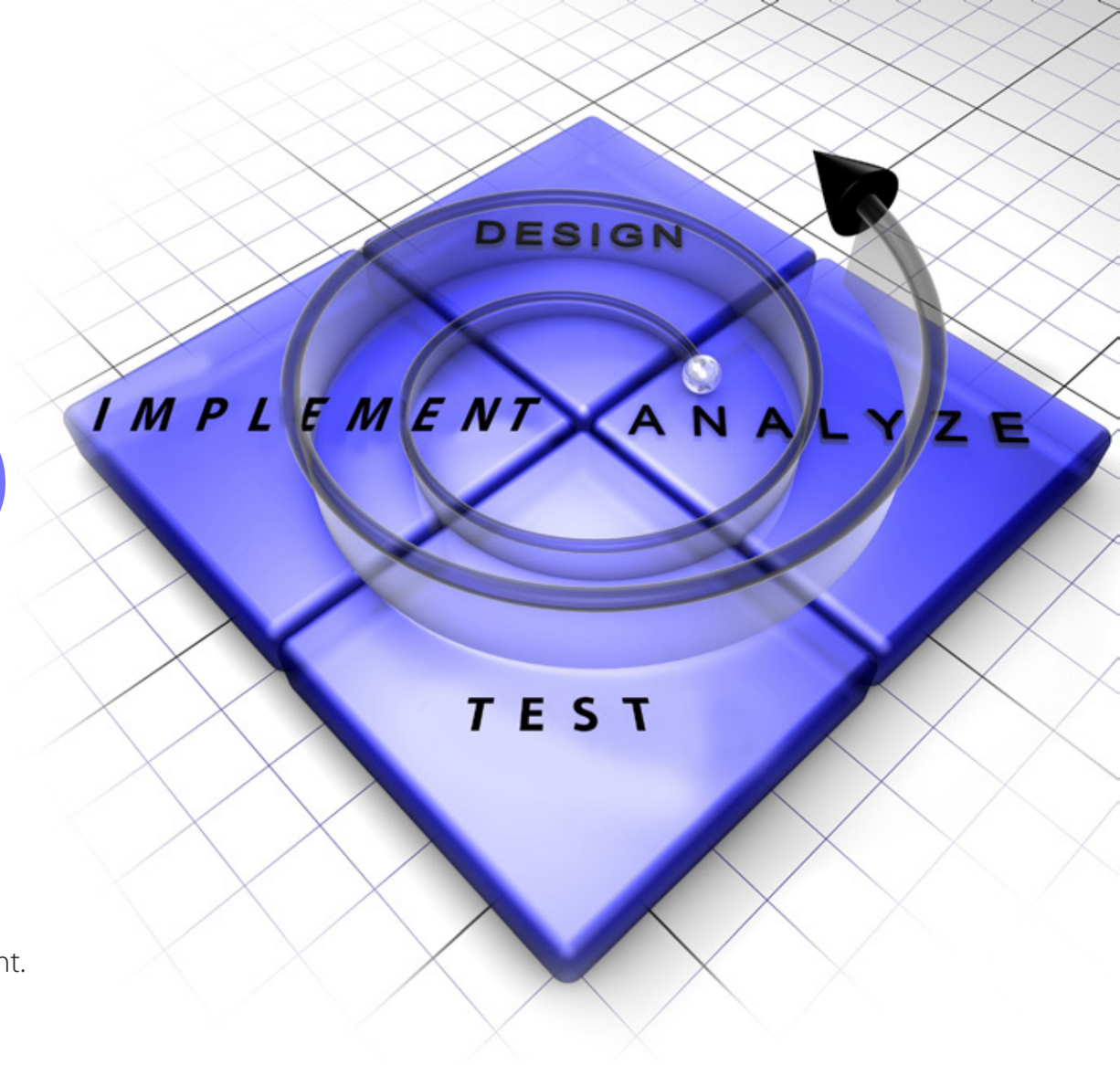


The process of software development has changed through the decades. The abstract nature of software makes it difficult to estimate, repeat and master in general. Although methodologies, tools and platforms come and go, people remain the constant in the equation. The agile approach is unique in that it focuses on the important aspects of the development process: the people and the artifacts — software and hardware. Agile encourages developers to be more iterative, team-oriented, self-improving and customer focused.

Another key to the agile process is its adaptability. No two companies, development teams, software products or customer bases are the same. The creators of the process knew this — and that's why agile shines when compared to other, more rigid, methodologies.

Perhaps more than in any other domain, this adaptability is key when it comes to embedded systems development. The unique challenges of embedded development demand a process that can conform to its constraints and requirements. Agile can be adapted to the embedded developer's needs; in turn, agile development can also change and improve the embedded systems development life cycle.

What Is Agile?



Agile development is a group of software development methods based on iterative and incremental development. With agile development, requirements and solutions evolve through collaboration between self-organizing, cross-functional teams.

Agile development promotes adaptive planning and evolutionary development and delivery. It also encourages rapid and flexible response to change.

Created by a group of developers in 2001, agile builds on key lessons from past development processes, such as waterfall, spiral, CMMI and extreme programming (XP): Frequent iteration, improvement and communication are paramount.

The Agile Manifesto and Principles

The Agile Manifesto and its 12 core principles define agile development in only 68 words:

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

- *Individuals and interactions over processes and tools*
- *Working software over comprehensive documentation*
- *Customer collaboration over contract negotiation*
- *Responding to change over following a plan*

That is, while there is value in the items on the right, we value the items on the left more. Agile helps developers avoid two big pitfalls common to other processes:

- *Integration nightmares that occur at the end of the development cycle*
- *Finding out that your customers' needs weren't met after the software is released*

As a result, software developed using the agile process is almost always in a state that's ready to be released with the most up-to-date, customer-approved features.

Applying Agile for Smarter Development

Agile adoption continues to grow throughout the global software development community.

Figure 1. UBM Tech survey of business technology professionals at companies with 100 or more employees who use source code management (SCM) tools, January 2013

A recent UBM Tech survey of business technology professionals (developers, testers and managers) shows that out of those polled only 5 percent had no plans to practice agile at the time of the survey (see Figure 1).

Agile isn't just about getting things done quicker, it's about being more efficient and effective, delivering higher-quality artifacts, creating accurate estimates, measuring effectiveness and improving team collaboration.

In general, organizations adopt agile because it:

- Results in higher quality
- Helps address integration issues earlier in the development process
- Provides a feedback loop through rapid iteration
- Allows requirements to be refined earlier rather than later
- Helps everyone deal with change
- Enables working software with new features to be released more often
- Ensures that everyone works at a sustainable pace

- Increases productivity (due to gains in efficiency, not hours worked)
- Adapts to unique and diverse development projects

The flexibility of agile is probably most evident with its adoption in the embedded systems development do-main, which involves unique challenges not seen in typical IT projects. Embedded agile practitioners need to deal with constraints that their enterprise counterparts may never encounter. While the principles of agile can accommodate embedded development characteristics, agile is turn-ing embedded development upside down. ■

Agile Development Characteristics

The goals of agile are straightforward: collaborate, deliver new functions in frequent increments and improve continuously. To achieve this, some formal but adaptable processes are involved.

First, there are primary roles for people on an agile team:

- **Team member** (agile practitioner): responsible for the end product
- **Team lead** (agile manager): leads the team to achieve success
- **Stakeholder:** the project sponsor, the end user, IT staff and so on
- **Product owner:** represents the needs of the user

The agile process begins with release planning, in which the release schedule is divided into iterations (typically about two to four weeks in length), and work items are defined. However, as with most things in agile methodol-ogy, this time frame is flexible, not rigid. The length can be changed according to the specifics of each project and other embedded development challenges.

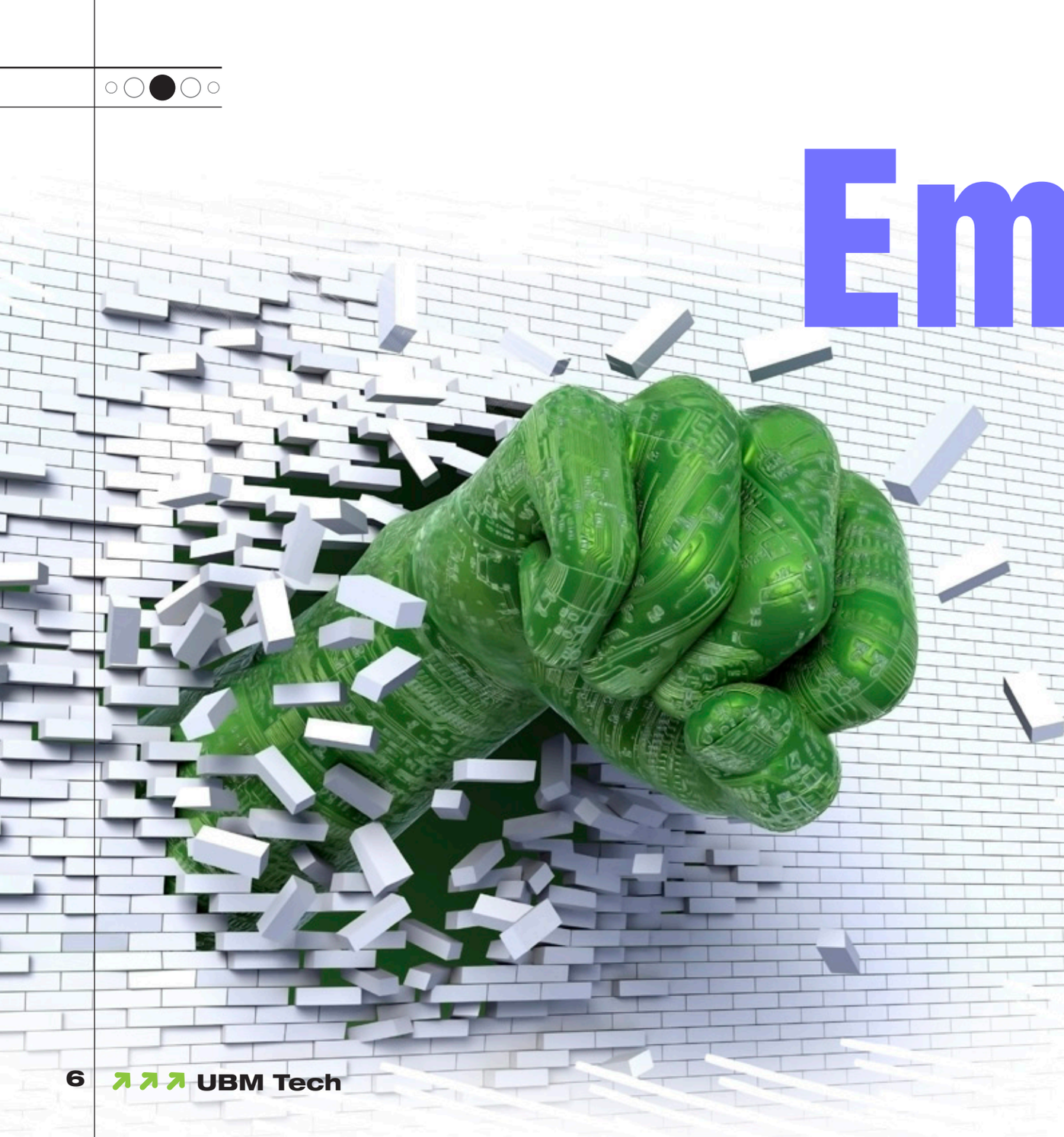
At the beginning of each iteration, often called a sprint, planning determines which work items will be part of that iteration. A daily planning session, often called a *stand-up meeting*, helps team members quickly collaborate and coordinate.

Work items are often called *stories*, mainly because of the format in which they're described, which may be further broken down into smaller tasks, and then placed into a list called the backlog. Stories that are planned for the current sprint go through a group estimation process, marked as in-process, and are then assigned to a team member for completion.

Along with suggested estimation practices, agile teams continually measure their progress, track their productivity — referred to as velocity — and improve their process with each iteration, during the iteration retrospective meeting. The agile process also supports practices such as the use of modeling, test-driven development, automated unit test-ing, and continuous integration and deployment.

To realize agile development, agile principles must be in-stantiated in a development process. Agile processes may vary between teams and organizations according to their unique challenges and may draw on a variety of imple-mentation approaches including Scrum, XP, Lean, Kanban, Harmony and modeling-based methods. The exact steps and terminology are suggestions, and can be adapted to the unique requirements and challenges of each team.■

Embedded Challenges



Embedded systems development takes on many forms and involves many challenges, not the least of which is the complication of custom hardware. Often, embedded developers have to hit software milestones before hardware is available, and then deal with hardware and software revisions. Vertical-market standards and constraints, regulatory requirements, and safety-critical and real-time systems considerations further complicate the process.

Hardware and Software Perspectives

Embedded developers need to identify and depend on technology specialists (mechanical, electronic, security, safety, reliability, and system and software engineers, to name a few) and consider supply chain constraints, manufacturing lead times and other hardware issues not encountered by enterprise agile developers.

"For the most part, enterprise developers only need to deal with software and requirements," explains Harry Koehnemann, director of technology at 321 Gang. "Servers and patterns, especially around Web applications, are well defined and rarely need to be customized. Embedded developers need to deal with new hardware designs and mechanical requirements and create brand new architectures to solve specific embedded problems on top of writing software."

Contrary to conventional wisdom, embedded development doesn't necessarily need all requirements to be defined up front. Because hardware may not be available, developers end up testing as features become available, iterating often along the way.

"The first problem any embedded developer encounters is that you usually cannot test on the PC you're developing on," according to Martin Bakal, electronics industry offering manager at IBM. "Instead, you test in a lab, building and integrating regularly, and testing continuously."

Organizational Challenges

Embedded development often requires domain experts and specialists who work in different ways. Some experts may not be located with software developers, because they need to work in a lab environment, for example, or because they're located in other regions of the world. As a result, development teams may involve a community of engineers that are more distributed and remote.

Additionally, embedded development projects typically experience a large number of dependencies in terms of systems integrators and third-party suppliers.

Vertical Market Challenges

Embedded systems development is often targeted to a specific vertical industry such as health care, aerospace, industrial automation, automotive, logistics, telecommunication, manufacturing and so on. Each of these verticals has its own regulatory requirements, standards bodies and governance requirements — often in the area of safety.

Agile for Vertical Markets

Vertical markets introduce new demands and constraints on the agile process, and dictate how agile can be applied. Yet, the frequent releases and other efficiency benefits make agile an attractive model for embedded systems development in many industries.

Aerospace and other industries, for example, face constraints in the form of government rules and security clearance. "In terms of aerospace, [agile is used frequently in projects] working on classified programs, as well the smaller ones," according to 321 Gang's Koehnemann. "Often agile is embraced

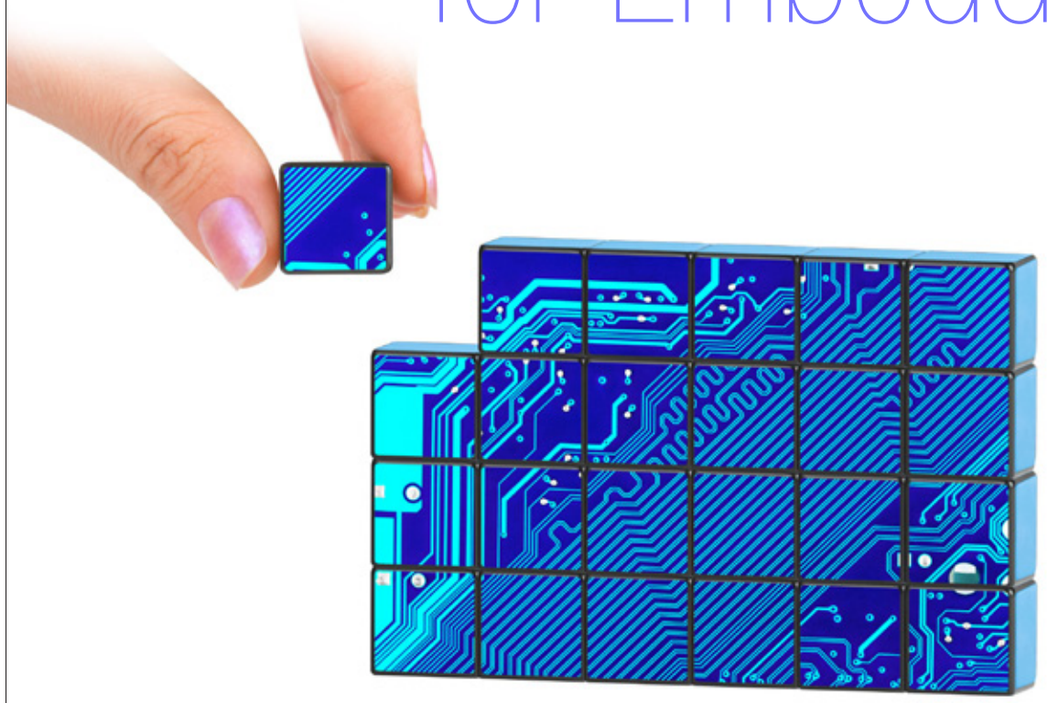
because there's greater demand to get what they're doing out into the field. The shorter release cycle of agile is helpful to meet these real-world problems. In terms of planning and collaboration, however, classified is always slower."

In the healthcare industry, similarly, medical device requirements need to be formalized, go through sign-off, and get translated into design documents due to regulatory issues. However, there are ways to stay agile. AAMI TIR45 is a guidance document that provides advice on the use of agile practices in the development of medical device software.

The discipline and other techniques required in embedded development can often apply in other development domains as well. For example, in financial services, security is a key consideration. Bruce Powel Douglass, chief evangelist for IBM Rational and author of the book *Real-Time Agility*, described how a bank used safety-critical requirements to drive development of a new system. "We helped [the bank] apply defensive development techniques and other safety-critical methods to their enterprise IT infrastructure," Douglass says. The approach helped [the bank] safeguard its infrastructure as well as meet banking regulatory requirements.

Adapting agile to these constraints can be a challenge. For example, whereas small changes in requirements discovered in successive iterations are straightforward in projects that aren't subject to government regulations, the need to obtain regulatory approval for changes in the systems domain may affect just how agile an embedded development organization can truly be. ■

Making Agile Work for Embedded Development



Embedded development involves a process, whether it's formal or informal. In fact, process choice can determine the success of an embedded development project. Some experts suggest that previous development methods may have led to a focus on the wrong questions.

"CMMI resulted in heavy dead-tree libraries of procedures," say IBM's Douglass. "It asks questions like, 'Did we do what we said?' instead of, 'Are we doing the right thing?'" Douglass tells the tale of a company that had adopted the ISO

9000 standards for quality management. "We looked at the metrics and found that projects under the new procedures were 25 percent less efficient with no measurable difference in quality. So they were less efficient and had less to show for it," Douglass recalls.

There are clear advantages to using agile for embedded systems development. Even organizations that don't specifically set out to implement agile may already be unknowingly practicing many of its tenets.

According to 321 Gang's Koehnemann, many embedded developers already work iteratively. "Most of the systems development organizations I've worked with have two goals," Koehnemann says. "First, hit a specific date. Second, attain a predictable outcome with each release. Agile helps to deliver on these two goals."

Projects derive the biggest benefit when agile is used from the beginning, according to the experts interviewed. Agile changes the thinking from components and developers to features and customer benefits. Incrementally building in features as the project moves along removes the big integration effort and associated problems often seen at the end of embedded development.

The experts also agreed that agile helps to reduce cost and risk through its iterative process. "It removes needless documentation, and saves costs through improved quality," says Bakal. "Without agile, interface changes would often occur late in the project, introducing many costs. With agile, this doesn't happen anymore."

Modeling and the Embedded Agile Process

Modeling, as part of the agile process, can help to streamline the turning of requirements into product designs. Models can also be a useful substitute for hardware in product development environments where the hardware may not be available to meet agile software development timescales. For example, modeling and simulations can prove out designs and software before actual hardware is built.

"Even source code is a model," says IBM's Douglass. "It's not fundamentally different from UML, but somewhat more removed from the problem domain. The problem is that there is a gap between code and the business problem. For instance, how do you turn the requirement, 'figure out where this plane is in the air' into code? You model the problem domain with state, behavior, classes, then turn that into code," explains Douglass.

According to Douglass, there are three camps regarding agile modeling:

1. Those who say they don't need it because their system is very small
2. Those who embark on a "lightweight modeling" effort to aid in discussing a system's design conversion (however, it's difficult to determine the correctness of a system without a precise view of what's involved)
3. Those who use high fidelity, detailed models that can be verified and reasoned about

According to Jordi Manzano, R&D deputy manager at Diagnostic Grifols, S.A., building a simulator is one of the first work items in an early sprint. Diagnostic Grifols makes

embedded systems that automate the analysis of blood samples to check the compatibility of donors and recipients. Company hardware and software engineers use IBM Rational DOORS and IBM Rational Quality Manager to manage an embedded agile development process, meeting FDA compliance while tracking tests to requirements.

Manzano credits agile with helping enable a more parallel, efficient development process. Because of agile, Manzano says, Diagnostic Grifols software developers don't need to wait for hardware, and continuous testing makes the transition to new hardware a smoother, more robust process.

Modeling is most helpful when you use the right tools. Static models quickly become obsolete as requirements and code change. Instead, IBM's Douglass advocates an agile approach to modeling that involves a central repository of modeling data, where models and working source code are the output. This way, when the model data is changed, every artifact is updated together.

Continuous Integration and Change Management

The iterative nature of agile has driven adoption of continuous integration (CI) processes and tools in software development. With embedded systems, hardware development needs to be considered. Additionally, there's the need to build and incorporate cross-compilers and tool chains into traditional CI tools.

Bakal and Douglass agree that agile encourages embedded developers to use modeling and simulation to test continuously. Early involvement with customers in each sprint also results in more rigor.

"Engineering was less rigorous than it should have been in the past," says 321 Gang's Koehnemann. "Agile enforces the rigor needed. Model-based engineering is the best way to ensure that you understand the behavior of your designs before committing hardware and software development to it."

The right tooling is critical to implementing agile in the embedded domain. Some tools may need to be adapted to the embedded world. A better option is to examine tools specific to the embedded domain.

"Something needs to make it all flow and work together," IBM's Bakal says of continuous integration. "This includes hardware-specific tools. They have to somehow work together, so you need an ecosystem to make it work where you can plug in the tools you want."

Each group can't go in its own direction and pick its own tools, cautions Bakal. But different groups can mix and match features of a common set of tools. "Continuous integration [platforms] such as IBM Jazz can integrate this way," Bakal says. (For more about IBM Jazz, see jazz.net.)

Agile helps better manage the changes that are inevitable, according to the experts interviewed. Finding them sooner, before committing resources, helps to save money. It also creates more signs of progress, more accurate time estimates and better visibility into schedule slips.

"In the past, the functionalities needed to test the hardware were not ready as the hardware became available, which led to delays," says

Diagnostic Grifols' Manzano. "With agile, we start the user application in parallel with the manufacturing and service application, which handles the hardware, makes the devices move, and other actions. This is to allow the hardware engineers to continuously test prototypes early on. It enables iterative and early integration. The road map is aligned with hardware milestones, and each release is developed using agile."

Before agile, Manzano says, only hardware had visibility. "You actually see engineers with hardware and mechanical drawings. With software, all you see are lines of code, maybe," he explains. "Now with Scrum, I know what I have with each sprint. I can run the software, log in and use it. Before agile, integration was the 'stuff-it-all-in-at-once' time."

For embedded agile, innovation can represent new ways to address the unique requirements and constraints of the systems being built. For instance, companies building products in the highly competitive consumer space have tight delivery timescales to meet or they risk losing the market. Agile has helped these organizations meet their innovation-driven timelines with predictability, according to 321 Gang's Koehnemann.

"In the automotive space you cannot miss your deadline for the next model-year car," Koehnemann says. "Many embedded automotive projects are moving to agile to be more predictable in terms of what goes into each release. As a result, auto makers can focus on the innovative features that they know will make it into the next model year."

The Management Perspective

Classic agile takes into consideration managing and practicing the agile process. Agile aims to build the knowledge of the group as a whole, avoiding the creation of silos and specialists. However, embedded systems development typically brings a more compartmentalized, distributed group of people with specialties that can't be avoided.

"While it may not be possible to have individuals who are experts in both hardware and software development, there are other areas [in which] silos can be avoided, such as customer interaction and requirements," Bakal says. "In terms of customer pain points, because of agile, all of the parties now share this."

Another goal of agile is to get adoption throughout the organization. "In R&D and product development, adoption is growing. This includes development, QA, testing, and production support. But it's not moving much outside of these departments," Diagnostic Grifols' Manzano observes.

321 Gang's Koehnemann sees a similar phenomenon. "In the embedded community, developers often try agile out of necessity. It's a natural solution based on how they work. The developers are agile, but often product management is not." Koehnemann went on to mention that in terms of the rest of the organization, there's room to add customer satisfaction, customer support metrics, and other success/failure metrics to measure agile process effectiveness beyond just velocity, burn-down and other development-centric metrics. ■

IBM Rational Agile Solutions

Through numerous client engagements and its extensive software development experience, IBM has identified three imperatives for a successful transformation to agile in embedded software and systems development.

Traceability Across the Life Cycle

Capabilities must be tightly linked across all functions in the engineering life cycle. For example, requirements must connect to change requests and to models and then to workflow tasks and on to verification and validation procedures and even into the operations, maintenance and product evolution phases. IBM solutions for embedded product and systems development enable traceability to be effectively created, maintained and utilized across multiple engineering disciplines and tools, extending to tools and data sources provided by our partners and other third parties through an open, linked life-cycle data approach.

Access to All Engineering Information

The old ways of import/export and emailing information around won't cut it — they aren't fast enough or agile enough to meet today's needs for rapid innovation. And a single database or repository isn't practical in the rapidly evolving world of competitive product development. Developers must be able to query and organize information from across the life cycle and across the variations in a product line so that everyone has a holistic view of the software and systems they are building.

IBM has developed a capability to enable development teams to access engineering data that delivers:

- Clear views of relevant and related development data
- Analysis of data that shows the impact of changes
- Data organization that provides context for queries and decision making

Collaboration Across Engineering Disciplines

Embedded agile developers cannot work in isolation from other team members and other engineering disciplines — not if companies want to survive and prosper. Teams must collaborate in real time and work together in new ways, even if they use different tools from different vendors. Change requests and their impacts may affect multiple developers and disciplines, and everyone must be kept up to date. Faster product innovation cycles mean more virtual, model-based development and closer coordination between everyone.

IBM is a leading supporter and contributor to OSLC, Open Services for Lifecycle Collaboration. OSLC (see: www.open-services.net) is a community of tool vendors, systems integrators and end-user organizations that are working to standardize the way that life-cycle tools can share data with one another. OSLC defines a simple, Internet-inspired way for tools to share and link data across domains such as requirements, architecture, quality and testing, and change management.

To find out more about IBM's solutions for embedded agile development, visit: *http://www.ibm.com/software/rational/agile/embeddedagile*

Building a Smarter World: **Agile Embedded** Success

The right tooling solutions can make the difference between success and failure when adopting the agile methodology. Agile is a process that helps organizations become smarter as they learn from each iteration. To do this effectively requires a tooling solution that can operate with minimal friction across the life cycle. It's no longer sufficient to pick up tools from various sources and expect to have an effective, integrated solution.

IBM Rational solutions can help embedded development teams become more agile with:

- Agile planning and execution, allowing teams to flexibly plan and manage delivery
- Life-cycle capabilities, including requirements management, model-based architecture and design, quality management, and change and configuration management
- Integrated, customizable agile processes that can support industry-specific standards compliance
- Extensible solution configurations to address the needs of key industries, including aerospace and defense, automotive, electronics and medical devices
- Process guidance and assets as extensions to the Harmony process, including specializations for aerospace, automotive, and medical domains

[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* ■