Agile Product &
Project Management

Vol. 12, No. 11

Executive Report

CUTTER CONSORTIUM

Is Agile Shortchanging the Business?

by James Robertson and Suzanne Robertson,
Senior Consultants, Cutter Consortium

Agile methods provide a very efficient way to develop software. But efficiency is not the point. As we explore in this *Executive Report*, the point is that in the rush to experience the virtues of this effective development method, and the excitement of the surrounding publicity, an important question is left unanswered: *Does efficient software development (read "agile") necessarily bring a real advantage to the owner of that software?* A growing number of our clients are concerned about business value and that the software they take delivery of is not fully exploiting the potential value. To put it another way, there is real business value to be had, but the software development process alone does not deliver it. Clients complain about the lack of innovation coming from agile development teams, and they report that their new software is often not that different from the previous incarnation of the functionality — a few extra bells and whistles, yes, but not the breakthrough implementation that delivers a significant business advantage.

Access to the Experts

Cutter Consortium, a unique IT advisory firm, comprises a group of more than 100 internationally recognized experts who have come together to offer research, consulting, training, and executive education. These experts are committed to delivering top-level, critical, and objective advice. They have done, and are doing, groundbreaking work in organizations worldwide, helping companies deal with issues in the core areas of software development and agile project management, enterprise and business architecture, business and technology trends and strategies, innovation, enterprise risk management, metrics, and sourcing.

Cutter offers a different value proposition than other IT research firms: We give you *Access to the Experts*. You get practitioners' points of view, derived from hands-on experience with the same critical issues you are facing, not the perspective of a desk-bound analyst who can only make predictions and observations on what's happening in the marketplace. With Cutter Consortium, you get the best practices and lessons learned from the world's leading experts — experts who are implementing these techniques at companies like yours right now.

You can tap into this expertise via print and online research services and journals, mentoring, workshops, training, and consulting. And by customizing our information products, training, and consulting services, you get the solutions you need while staying within your budget.

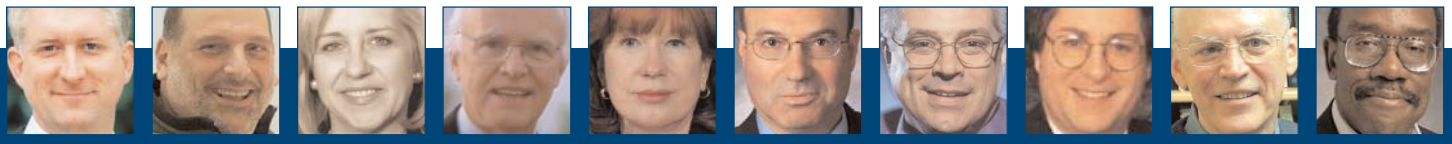
Cutter Consortium's philosophy is that there is no single right solution for all enterprises, or all departments within one enterprise, or even all projects within a department. Cutter believes that the complexity of the business technology issues confronting corporations today demands multiple detailed perspectives from which a company can view its opportunities and risks in order to make the right strategic and tactical decisions. The simplistic pronouncements other analyst firms make do not take into account the unique situation of each organization. This is another reason we present the several sides to each issue: so you can determine the course of action that best fits your unique situation.

Expert Consultants

Cutter Consortium products and services are provided by the top thinkers in IT today — a distinguished group of internationally recognized experts committed to providing top-level, critical, objective advice. They create all the written deliverables and perform all the consulting. That's why we say Cutter Consortium gives you *Access to the Experts*.

For more information, contact Cutter Consortium at +1 781 648 8700 or sales@cutter.com.

The Agile Product & Project Management Executive Report *is published by the Cutter Consortium, 37 Broadway, Suite 1, Arlington, MA 02474-5552, USA. Tel: +1 781 648 8700; Fax: +1 781 648 8707; Email: service@cutter.com; Website: www.cutter.com; Twitter: @cuttertweets; Facebook: Cutter Consortium. Group Publisher: Kara Letourneau, Email: kletourneau@cutter.com. Managing Editor: Cindy Swain, Email: cswain@ cutter.com. Print ISSN: 1946-7338 (*Executive Report, Executive Summary, *and* Executive Update*); online/electronic ISSN: 1554-706X.*



Rob Austin | Ron Blitstein | Christine Davis | Tom DeMarco | Lynne Ellyn | Israel Gat | Tim Lister | Lou Mazzucchelli | Ken Orr | Robert Scott

Cutter Business Technology Council

Is Agile Shortchanging the Business?

THIS ISSUE'S AUTHORS

James Robertson
Senior Consultant, Cutter Consortium

Suzanne Robertson
Senior Consultant, Cutter Consortium

IN THIS ISSUE

Let's start with the idea that software is an enabler of work. We build software to help people do their work, to make work more efficient or cheaper, to provide better information, to allow people to do things that the slowness of human processing would not allow them to do, to comply with the law, and, in some cases, we build software because it provides its consumer with entertainment.

To make sense of this, we need to say (you already know this) that there is a difference between a *problem* and a *solution:* the problem exists in fact, but the solution is something constructed to solve the problem. Given that the solution is man-made, we can say that there can be a difference between a business need and the implementation meant to satisfy that need. The need is absolute, but if the need is not correctly understood, then it is possible — indeed likely — that the wrong solution will be produced.

We can also say that when a solution is proposed (or delivered) and the need it is satisfying is not understood, or has not been explicitly stated, then it is highly likely that the solution will not satisfy the real need.

So far these are historical facts, and Cutter readers are unlikely to dispute them. After all, it is plainly obvious that you cannot build serious software until you have understood the need that it is satisfying.

The need we are speaking of here is that of the owner's work, and more specifically, the improvement or benefit that the solution is intended to bring to that work.

THE WORK

There are many things we can do when we are using software. However, for the sake of simplicity, we shall refer to whatever it is that the software user is doing as "work." It might be a stretch of the imagination to think of listening to music or watching movies as work, but as we pointed out, the software must provide some assistance to some task that the user wants to do. So whatever it is that the user wants to do, let's call it

"work." *The work is the beginning point for all software development. This is because the software has only one purpose and that is to improve the work.*

We first have to consider the work from the point of understanding it. We also have to think about how the work fits into its wider context if we are to find a solution that best supports it. Understanding a piece of work, whether it is processing insurance claims, buying movie tickets, or guiding a surgeon's instruments during microsurgery, means considering what else is happening besides someone sitting in front of a computer. How does this piece of work fit into the overall organization? What is the business problem as distinct from the software solution?

The work is the beginning point for all software development. This is because the software has only one purpose and that is to improve the work.

As a well-known example, Apple's iPod has been a raging success due to its combination with the iTunes Music Store. Instead of simply concentrating on the device they wanted to sell, Apple designers looked instead at the work of listening to music. By seeing how music fans had to rip their CDs before installing them on the iPod, Apple realized that the work is not just pressing buttons on the device but also includes the acquisition of music. So Apple opened the iTunes Music Store with its millions of songs. Music fans could now simply and conveniently download whatever they wanted to listen to. Moreover, music lovers also shared their music with a few close friends, usually by listening to each other's CDs. So Apple offered household sharing to make this convenient and thus provided previously undreamed of benefits to their customers. The success of Apple in the music arena attests to the value of taking this wider view of work.

OWNERSHIP

Regardless of the work being done, any software that is developed to support that work will be owned by somebody.

In the case of personal computers, a customer usually buys software off the shelf at the local software store, or via downloading. In this case, the buyer becomes the owner, and at the risk of confusing this issue, he or she also becomes the user of the product.

Alternatively, the software could be custom-built for an organization. In some cases, the software's owner is also the organization that is doing the building. Sometimes the software is built on-demand by a software house. Either way, the software comes to be owned by the organization that will use it. There are, of course, other ways that people and organizations can come to own software.

Keep in mind that when we speak of ownership here, we are not referring to the "product owner" role on most agile teams. While the product owner role is responsible in part for the delivery of the product, he or she is doing so on behalf of the real owner: the person or organization paying for the delivered product.

We contend that it is the owner of the software — not the user and not the developer — who is the person to be considered here. Of course, we have to make the product acceptable and enjoyable to its end user, but this is so that the user will use it effectively and willingly, which in turn benefits the owner. As discussed earlier, the work belongs to the owner, so it is he or she that must get the benefit of the end product. The implication of this is that if we are going to build software, we must build it in such a way that is optimally valuable to the owner.

AGILE TECHNIQUES

There are various agile techniques, mostly based on the Agile Manifesto and mostly making use of incremental delivery of the product via iterative development. The manifesto includes the line: "Customer collaboration over contract negotiation," which we heartily endorse. However, we have some serious doubts about the contribution that can be made by the customers on agile software development teams. Agile techniques are very efficient for developing software and indeed effective for delivering software, but through our observations, we are forced to question whether the software is always valuable.

As mentioned previously, most agile teams have a role called "product owner" or some equivalent. The product owner is intended to represent the business point of view and is responsible for generating the requirements in the form of user stories. In many cases, the product owner is actually "somebody from the business." Herein lies the problem.

The product owner generating most of the user stories is unlikely to be trained to discover, prioritize, and communicate business requirements. Instead, the product

owner generates user stories that focus on the details of a software interface. We will say more about this later.

There is an empty chair on agile teams — indeed in many software development teams. This empty chair should be occupied by someone who can take an holistic view of the work and how the product will improve that work. This someone should ensure that the software being developed meets the true (and future) needs of the software's owner. This requires the skill of investigating the work that the business is doing and identifying which parts of that work would most benefit from improvement. And if that is not enough, this someone should be coming up with innovative changes to the owner's work and innovative ways to address the resulting business needs.

The product owner on an agile team is responsible for providing the business view when setting user stories or providing requirements. However, this is trying to channel the whole business knowledge through the wrong person. Our experience tells us that in trying to maintain velocity, it is difficult, if not impossible, for someone closely connected with the day-to-day activity of the development team to take a dispassionate view of what the business is doing and what the business actually needs. The result is that the development team pays less attention than it should to deriving innovative solutions, thereby ultimately disappointing the owner when the software is delivered.

BUSINESS NEEDS

Businesses are large and complex. They are made up of a large number of processes that link together to form a coherent whole. It is the coherent whole that we are concerned with and how each part of that whole contributes to the organization's goal. One unfortunate aspect of modern business is that it is often difficult to get a complete picture of how all the parts fit together. In some cases, this means that the organization performs poorly because its business processes are not acting congruently with the common goals.

Farsighted businesses understand that it is necessary to take a *systemic view*. This means considering how all the pieces work together and how a malfunction in one part of that can have a knock-on effect in a distant part. Alternatively, an improvement to one part may have an unintended detrimental (but hopefully beneficial) effect on another.

Figure 1 is an illustration of how an organization is made up of activities that are connected to other activities — some of which are carried out by other organizations. The activities that comprise an organization — departments, branches, and areas of responsibility — are each in turn comprised of many smaller activities that once again have many interfaces with other activities.

Consider a part of an organization composed of people carrying out business procedures, pieces of software, pieces of hardware, machines, telephones, and so on. Each of these pieces is a specialized part of the overall organizational system, and each requires specialized knowledge. A change to any part of the overall system has the potential to affect many other parts. Unless someone is taking a systemic view of the organization, the likelihood is that many well-intentioned changes will not maximize the value of the business investment, and in many cases could cause damage.

Failure analysis[1] is the technique of measuring how much of the organization is not performing as it should, how much it costs in damage to the organization's service or product, and how much it costs to repair the malfunctioning process. This analysis is sometimes little more than measuring and monitoring complaints. But given the propensity of modern businesses to provide poor customer service, it is highly likely that many customers do not bother complaining when things do not go as they wish. In any event, businesses should be aware of what is failing and what those failures cost.

Taking the systemic view of the organization and doing failure analysis is one basis for determining where the

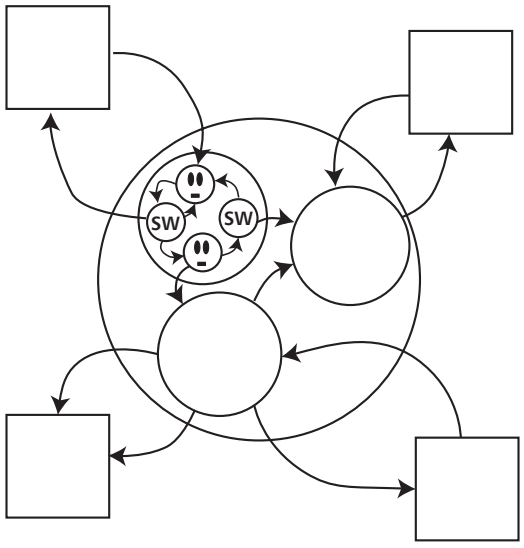


Figure 1 — An organization is made up of many pieces, each of which is connected to other pieces, including other organizations. In turn, each piece is made up of many pieces, and so on.

development team should make improvements and innovations that will deliver the maximum business value.

BUSINESS VALUE

Any investment that is made to change a business should have a demonstrable value to the overall business system. More and more of the changes made to a business involve the delivery of software. If the delivered software is to be valuable to the business — and simply calling it "customer value" does not ensure that it does — then the development team has to be conversant with the business and its intentions.

Assessment of business value relies on a consistent way of talking about the components in which you plan to invest. At the very least, you need to be able to articulate the business goal for making the investment, the scope of the affected work, and the stakeholders (both business and technical) who are affected by this investment. The primary concern is that everyone who is concerned with determining the value has a common language for discussing it. A shared knowledge model is a useful vehicle for building a common language.[2] Figure 2 is an example of a requirements knowledge model that can be used as a communication vehicle for both business and technical people.

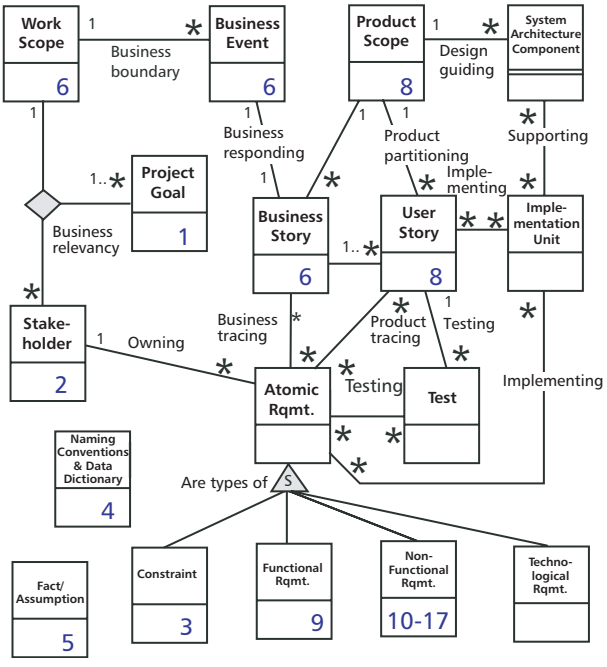


Figure 2 — Your own knowledge model might use a very different language. Regardless of your terminology, providing everyone has exactly the same understanding of each term, you can discuss the pieces, assess their value, and determine priorities.

Each box on the model represents a class of knowledge and is accompanied by a definition of precisely what it means to the people who own it. For example, if we refer to the definition of what the owners of this knowledge model mean by the class of knowledge entitled "Work Scope" we find the following:

- **Work Scope**
  - *Purpose* — defines the boundary of the investigation of the work that is necessary to discover, invent, understand and identify the requirements for the product.
  - *Attributes* — adjacent systems, input data flows, output data flows, work context description.
  - *Considerations* — should be recorded publicly as our experience is that it is the most widely referenced document. A context model is an effective communication tool for defining the work scope.

The important point is that everyone who is concerned with the business system has a common way of referring to all the different pieces. Your own knowledge model might use very different terminology. That is not important; the point is that you need to have and use consistent terminology within your own organization. This enables you to have conversations such as the following:

- **Business analyst:** "Of the seven input dataflows that we have identified and analyzed on the work scope, the business has confirmed that numbers three and six provide the highest value contribution to meeting the project goal defined by the business."
- **Developer/product owner:** "Have you done any work on the business stories for three and six?"
- **Business analyst:** "Yes, here are the two business story cards along with a first-cut data dictionary of the terms the business uses. A rough value analysis indicates that a solution to business story three will deliver marginally more immediate value."
- **Developer/product owner:** "OK, let's go through these together and then we will brainstorm the possibilities for user stories to support this business story."
- **Business analyst:** "Good, I'll call you tomorrow and see if we can make a decision."
- **Developer/product owner:** "We have identified four user stories for the business story number three. Here's a sketch model of the resulting product scope.

We could deliver software to support this in the first sprint, see how that goes, and then decide on the details of the next iteration."

- **Business analyst:** "Sounds good, let me know what else you need and I'll do whatever I can to get questions answered quickly."

This leads us to a discussion about the role of the product owner on an agile team.

PRODUCT OWNER

The role of the product owner, or whatever the customer representative is called on your team, is insufficient. There is a need for someone whose interest is not to produce software, but to understand and communicate a holistic understanding of the business that will use the eventual software.

The product owner is unlikely to be trained at discovering and writing business needs. The user stories are meant to represent the requirements from the point of view of the business. However, the very title "user stories" indicates that these are generally written to be a solution to the problem from the point of view of one user, who is being represented by the product owner. For someone to write solutions to business problems without first stating the problem is tantamount to asking an airline passenger to design the aircraft. The passenger might be good with basic things, such as more leg room, toilets, better elbow room, and so on, but is unlikely to be able to understand the complete needs of the aircraft — or all the work that the aircraft has to support. If we continue the analogy, then asking passengers to design an aircraft would mean that we would have excellent seats and toilets and galleys, but how they would fit together to form some flying craft remains somewhat of a mystery.

The problem manifests itself when we consider that, as many authors claim, the product owner writes the user stories. This implies that it is the product owner who is responsible for translating business needs into stories, and it is these stories that are the basis of the requirements for the software to be developed. This situation is illustrated in Figure 3.

Consider what is happening here: the product owner is being asked to understand a business, its needs, and aspirations; compose a collection of features for a software product that will improve the business; and, at the same time, come up with a set of features that will result in a desirable customer experience. It is not impossible for "somebody from the business" to have the necessary skills to do these things, but it is unlikely. Let's look at what is needed.

UNDERSTANDING THE BUSINESS

This is a task that is traditionally assigned to a business analyst, systems engineer, or someone with a similar title and training in understanding the business processes and data. This is a fairly specialized task, and we have seen over recent years how the number of business analysts in industry has grown and continues to do so. In these tough economic times, organizations tend not to employ anyone but those most crucial to the continued functioning of the business. The growth of this role in organizations points to the need for people to analyze and understand the workings of the business; that is, "how the work works." The emphasis here is on *understanding*, not necessarily documenting.

It is not enough to go to business users and ask what they want. They simply cannot know. This is not due to any failing on the part of the user but the impossibility of knowing the aspirations of the business and how software will help the business achieve those aspirations.

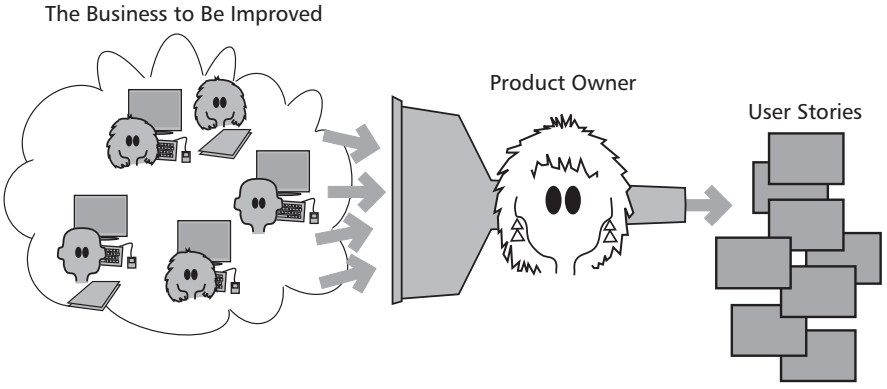


Figure 3 — The business and its needs are funneled through the product owner and translated into stories.

Businesses are complex organisms and to understand them enough to improve them requires more than a cursory study.

Translating Business Needs to Software Requirements

Before you can build any nontrivial software product, you need to be able to communicate the requirements to the developer of that product. Setting aside for the moment how you communicate the requirements, the need for clear and unambiguous requirements has been well and truly established over the decades that we have been building software.

It is the *means* of communicating the requirements that is the problem, not the need for requirements.

Some of the more excitable proponents of agile requirements lay claim to not needing requirements. They point to the fact that large requirements documents, completely written before development starts, are the downfall of many software development projects. These proponents appear to be saying that the solution is to get rid of requirements entirely. But the mistake here is to point at the document, and not its purpose. It is the *means* of communicating the requirements that is the problem, not the need for requirements.

This misconception has done a great deal of damage to agile. Naturally we still need requirements, but we definitely want to avoid large, unwieldy, procedural (and often irrelevant) documents. So let's stand back and consider what a requirement actually is and what form it would ideally take in an agile world.

A requirement is something the software has to do to support a business need. Indeed, if the software does not support a business need, then there appears to be little point in building it.

Requirements Are Not the Same as Solutions

A business need and the implementation of some technology to satisfy that need should be the same. That is, the solution fits the need. Unfortunately, this is not always the case. It happens for several reasons.

Sometimes the developer does not pay enough attention to the requirement, resulting in a product that is a close, but not perfect fit. In other cases, the requirement as it was written does not fit the need. In this event, no matter how faithful the implementation, the real need will not be satisfied.

But the most common reason for a misfit between the business need and the solution is that a solution is proposed without the proposer ever discovering the real need. That is, he or she has proposed a solution as a way of expressing the need. Which is the same as saying, "Here's my solution, but I can't tell you what the problem is."

Or to put that sensibly: if we do not know what the problem is, how can we possibly know what the solution is? Unless we can express the problem independently from the solution, we run the risk of never discovering the real need. With that in mind, let's look at user stories and how they are used, misused, and how they could and should be used.

USER STORIES AS REQUIREMENTS

User stories, usually originated by the product owner, act as the requirements for the product. These stories describe proposed functionality or features and are used to communicate with the developers about what needs to be built. The problem is not the concept of user stories, but the way in which they are used that brings with it unnecessary problems.

Our observations, consulting assignments with agile teams, and a lot of the literature, point to the fact that stories are often — alarmingly often — wrong. We understand that these stories are refined by a conversation with the development team, but it is inescapable that if the story starts out wrongly, then it is down to luck whether it ends up correctly.

> If you freeze an idea too quickly, you fall in love with it. If you refine it too quickly, you become attached to it, and it becomes very hard to keep exploring, to keep looking for better.
>
> — Jim Glymph, Gehry Partners

While we do not expect everybody to be skilled at understanding the real needs of the business, we do expect them to take the time to try. Instead, we see the undeviating rush to complete the sprint, without taking the small amount of time needed to ensure that the right software is being developed.

We started by saying the user stories are typically written by the product owner. This role is usually filled by "someone from the business." While this seems at first to be laudable, we have to consider that the holder of the role is part of the software development team. It is not exactly like letting the bricklayer decide where to locate the master bedroom in your new house, but it is not far from it.

By being part of the development team, the product owner can shift loyalties away from the concerns of the business and take on the concerns of the team. And the development team's main concern is to maintain the velocity of software production. Additionally, the product owner is a proxy for the real customers. In general, customer proxies don't work. Proxies are rarely trained to study the real business and, as a result, base their requests on assumptions and personal preferences with a focus on how an interface works rather than what business requirements it satisfies.

There is also the ever-present temptation of the product owner to be "self-referential." This term, coined by Alan Cooper, means that people describe what *they* want, rather than what will be best for and satisfy the greatest number of customers and users.[3] Self-referential also means that the end product might please the specifier, but not necessarily the organization as a whole.

User Experience

It is useful to keep in mind that today, a significant attribute of any software product is its user experience. This does not mean that we have to make a routine accounts payable system seem like an iPad to use, but we must ensure that using the software is pleasing to its users. And understanding what makes a pleasant user experience is beyond the realm of someone from the business. It takes a specialist, or at least someone with the appropriate training.

> Great experiences are about being human, and humans want to be surprised.
>
> — Eric Ryan and Adam Lowry, founders of Method

It is unlikely that a business user can design a good user experience. This seems to fly in the face of logic when it is the business user who is to use the software when it is delivered. Being a user is quite different from being a designer. A user-experience designer looks further than on just the work being done and considers the wider issues of making the work a better fit with its users and its technology. Good user experiences are designed from the outside in, not from the viewpoint of someone struggling with the daily task of getting all business work done.

Similarly, good user experiences are designed to start before any encounter with the actual software and go on during and after the usage of the software. Failure demand analysis — a necessary part of user-experience design — is unlikely to be done by someone who could well be contributing to the failure.

Just as we do not let airline passengers design the aircraft, why should we have business users design their business software?

"It Seemed Like a Good Idea at the Time"

Most authors agree that the general format of the user story is this: *as a [role], I want a [capability] so that [rationale for the capability].*

We suggest that the rationale is the most important part of the story. However, we are finding that many story writers, including some authors, omit the justification. This is both shortsighted and unproductive. For example: "As an energy user, I want to compare my energy use to others that have similar household demographics."

Good user experiences are designed from the outside in, not from the viewpoint of someone struggling with the daily task of getting all business work done.

The first question is "Why?" It might seem obvious to some, but the lack of a justification hides some important requirements. Why would someone want to compare his or her energy consumption to that of others? If he or she finds that they are using more energy, then they still have not gained the objective (the real objective was missed by the story writer). We might assume — and without a justification for this story we can do no more than assume — that the householder wishes to save money on energy bills; in which case, the real objective is to be told how to save money. A comparison might be mildly interesting but does not tell the householder how to reduce consumption. If the objective is to use less energy, then he or she really needs to be told what kind of reduction in energy use could be achieved by using alternative appliances, light bulbs, and so on. However, this information is only valid if there was a facility to tell the system about current consumption, types of appliances, and household routines. None of that was part of the original set of stories we encountered.

Is this story shortchanging the business? Without doubt — and always will when the writer does not bother to discover the real business problem. In the above case, adding the real justification would have revealed what the user needed, and an appropriate system could have been built. The story might have seemed like a good

idea at the time, but over time the code written from it will simply lie unused in a quiet corner of the user's hard drive. It won't be alone; it will join the millions of lines of unused code that also appeared to be a good idea but never had a justification.

Additionally, the justification provides documentation on the reasoning behind the requirement. Documenting the functionality of a system is indeed a waste of time — the code does it so much better. But the code contains few clues as to why it exists. For that you need the rationale to be part of the story. As we have seen above, lack of rationale often leads to an inappropriate system being built. But it always leads to frustration and time wasting when the system has to be maintained. Without knowing the reason for something, the maintenance developer will struggle to make the appropriate modifications.

> We know what it is doing, but we haven't a clue as to why it is doing it.
>
> — Anonymous maintenance developer

Lack of rationale often leads to an inappropriate system being built. But it always leads to frustration and time wasting when the system has to be maintained.

SYSTEMIC THINKING

Agile techniques, despite their advantages, tend to ignore systemic thinking in favor of the efficiency and convenience of developing software. User stories are prime offenders here, and the very name "user story" indicates why. By focusing on a user, the story naturally looks inward toward the software and not outward at the work of the business to be improved. As an example, let's have a look at a user story that came from a project to build a patient monitoring system. The story reads: "As a nurse, I want to input patient readings on a handheld device so that patient details are recorded quickly."

This is not necessarily a poor story, but it does show a lack of systemic thinking. The focus on technology indicates that the story is a solution, so we need to look a little closer into the problem it is trying to solve.

The story is written in the standard format: "As a … I want … so that…." Unfortunately, in this case, the format leads to an untruth. Nurses as a rule do not want to input anything, with the possible exception of the input of food and drugs and tender loving care, into their patients. The nurses might want the patient readings to be available, but they do not want to input the readings themselves. It would be valuable to investigate why the story suggests that the nurses care about whether or not the patient details have been recorded.

Consider the relationship between the nurses and the patients. The nurses want to spend as much time as they can helping their patients get better. The nurses need to analyze the readings of the patients' conditions so that their nursing takes the most appropriate action: call a specialist, arrange for a specific test, make a change to diet, and all the other things that a nurse needs to do in order to care for patients. To understand the real business need, one needs to ask why the patient details are necessary and what is the real reason behind the nurse's concern with inputting the details.

In the project that this comes from, the nurses not only asked for the facility to input the details, they also asked for handheld devices to do the inputting. Both of these requests covered up the real problem. Currently, the nurses recorded the patient details on a clipboard attached to the end of the bed (see Figure 4). Before the nurse finished a shift, he or she was supposed to input the patient details into the computer system. The problem was often that the nurses did not have time to do this input work. Either they had to work overtime to input the patient details, or they had to use time that they would otherwise be spent caring for the patients.

In fact, as you see in Figure 5, the nurses were asking for a solution to their current lack of patient care time problem based, naturally enough, on technology that they were aware of.

The nurses thought that, if they had handheld devices (specifically an iPad because one of them already had one), they could enter all the patient details when they were with the patient. They did not consider that they would still have to input all the details on their iPads into the computer system, and while there would be some improvements to be had by removing some of the double entry, they would have a piece of technology that they had to synchronize and that would divert them from time that they could have been spending caring for the patients.

Luckily the development team did not go ahead and build the solution that the nurses asked for. We say luckily because that solution was still not addressing the real problem. The business requirements analyst recognized that the real requirement is for the patients' details to be available to all medical staff without reducing the time the nurses spend nursing their patients. Armed with this understanding, the requirements

analyst, with input from technical stakeholders, investigated other solutions that would meet the requirement. The result was that there was other technology available for direct monitoring of most of the patients' details. That technology was already in use in other parts of this large hospital and so seemed to be the correct fit. Figure 6 illustrates how a large volume of the input was implemented by using the medical technology that was already available.

The nurses did not ask for this solution in their user story because they were not aware of its existence. Instead they covered up the real business requirement with a less-than-ideal solution. This is a very common problem in requirements work. The requirer (who is not a trained business analyst) asks for a personal perception of something that will solve a problem that he or she does not articulate. The resulting user story is one that does not necessarily solve the business problem and, if implemented, will often cause unexpected ripple effects into other parts of the business. This leads to the realization that user stories are a good tool for stating and communicating requirements, providing — and it's a big proviso — that the user story can be traced back to a real business need. In other words, we need to understand the business story before we can come up with appropriate user stories.

INTRODUCING BUSINESS STORIES

A business story, as the name suggests, focuses on a business need, within a scope of work, independent of the solution. You might know this kind of thing as an "epic," but we shall refer to it here as a business story because that name better indicates what it does. An

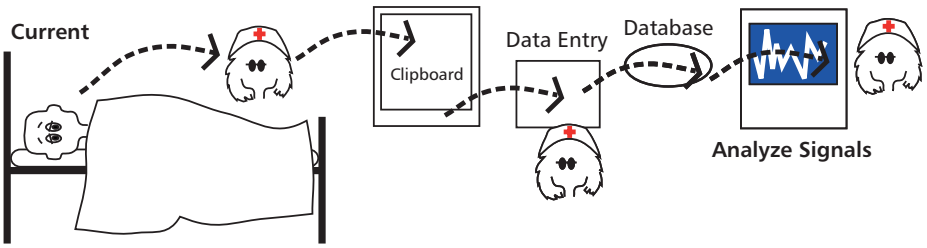


Figure 4 — The nurses are snowed under by the amount of data entry they have to do to transfer the patient details to the computer system.

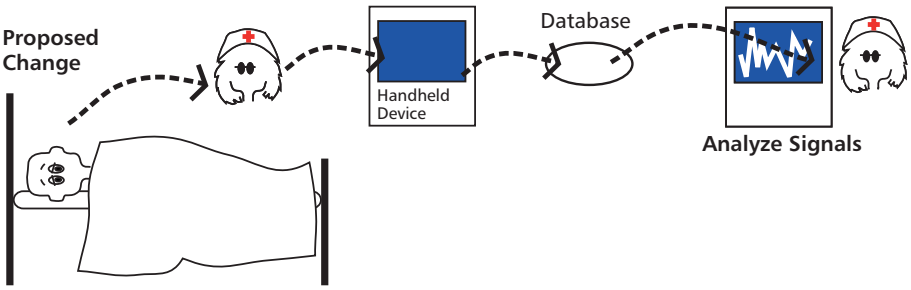


Figure 5 — The nurses asked for a solution based on known technology.

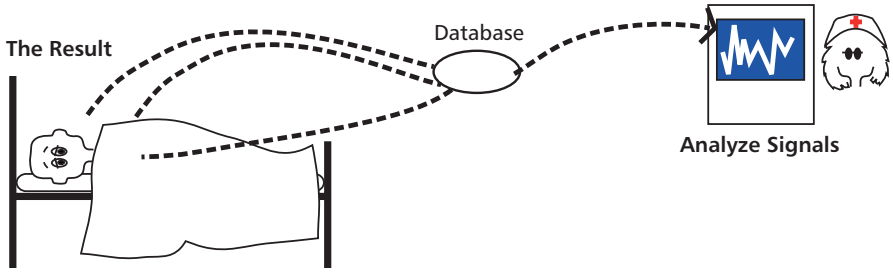


Figure 6 — The implemented solution connected the patient by sensors, and many of the patient details were captured automatically.

example of a business story within the patient monitoring scope of work is shown in the following examples of a nurses's response to a change in a patient's condition:

- Each patient has a number of key details that the nurse pays attention to. These details are things such as blood pressure, temperature, heart rate, and so on (full list on the nurse's patient details file for each patient).
- When there is a change in any of the key patient details, the nurse must decide on the appropriate treatment action (e.g., call the specialist, administer prescribed medication, or change medication).
- Any action that the nurse takes must be recorded as part of the patient's history.

This example of a business story is written using text. Depending on how you are working with stakeholders, and your own personal preferences, you might choose to represent this story as a business process model, activity diagram, rich picture, or any other form that suits the people in the communication.

The point is to be able to use the business story to get feedback from the appropriate business stakeholders and then to communicate with the stakeholders who will be responsible for coming up with the optimal solution.

The point is to be able to use the business story to get feedback from the appropriate business stakeholders and then to communicate with the stakeholders who will be responsible for coming up with the optimal solution.

From Business Stories to User Stories

Once a business story has been defined (i.e., you now know the real business work to be improved), then it provides the basis for deriving the most valuable user stories. In an agile team, the business story is used by the product owner as the basis for identifying the most relevant and beneficial user stories within the goal and budget of the project. For example, the goal of the patient-monitoring project is to free the nurses from administration work so that they can spend more time caring for the patients. Keeping this goal in mind, the following user stories can be extracted from the above business story:

- "As a nurse I want the system to record each patient's key details so that they are available for review and become part of the patient's history."
- "As a nurse I want the system to inform me when there is a change in a patient's key details so that I can decide appropriate action."
- "As a nurse I want the system to communicate emergency calls to the specialists so that I know that the patient's doctors have been informed."
- "As a nurse I want the system to record changes to a patient's treatment so that other qualified medical practitioners have up-to-date information about the patient's history."

Note that the above user stories leave the door open for the developers to come up with cost-effective ways for meeting the requirements in each user story. And each user story connects back to a business story whose purpose is to respond to change in a patient's condition in a way that maximizes the amount of time that the nurses spend caring for the patients.

Business Analyst Skills

Writing a business story needs to be done by someone who has the skills necessary to understand, analyze, and define a business problem. A business analyst is trained to observe and identify the real needs of the business and also to act as an intermediary between those in the business space and those in the solution space.

The business analyst does not have a vested interest in the existing business system, nor does he or she bring personal preferences to bear. Instead the analyst is trained to neutrally state a problem so that all interested parties understand the facts in the same way. As soon as the needs of the business are understood, then the solution stakeholders have a number of ways of meeting them. Each solution has different benefits and costs; the owner of the eventual solution makes a choice between. Thus, the business analyst's responsibility is to ensure that the owner understands the options for meeting needs well enough to make a reasoned choice.

NONFUNCTIONAL NEEDS

Agile methods pay too little attention to nonfunctional requirements. These are the requirements for such things as look and feel, usability, security, operability,

and so on. Some project teams seem to think it is acceptable to leave such details to the developers. And yet, here's the thing: provided that the functionality is met, then it is the nonfunctional requirements that determine the success or failure of the delivered solution. If you are scoffing at this notion, ask yourself why Apple is currently riding high with its array of stunningly easy-to-use and good-looking consumer products. The answer is nonfunctional requirements. Apple's products are functionally similar to others but are easier and more intuitive to use. Trust us, Apple does not leave it to the developers to decide on these properties of its products.

There needs to be someone who, once the user story has been chosen, has the responsibility for defining the most appropriate nonfunctional requirements to support the user story. For example, one of the nurses' user stories is: "As a nurse I want the system to record changes to a patient's treatment so that other qualified medical practitioners have up-to-date information about the patient's history."

Even when the system is receiving readings direct from sensors attached to the patient, there are still look and feel and usability issues. For example, can the nurse easily verify that the system is functioning correctly and that the flow of data is progressing as it should? Is the visual (or any other) feedback unambiguous, unmistakable, and easy to understand? Is the data secure and accurate? These are nonfunctional requirements and could well determine the success of the product.

Additionally, another nonfunctional requirement is performance. What volume of readings are we talking about, and how often are the readings transmitted? The story also talks about other qualified medical practitioners having up-to-date information about the patient's history. This raises the need for a security nonfunctional requirement to specify which medical practitioners may have access to which parts of the patient's history.

Agile teams are not in the habit of writing nonfunctional requirements or nonfunctional story cards. But these nonfunctional requirements need, somehow or other, to be discovered and communicated to the developers. Some teams we have worked with put together a checklist of nonfunctional requirements types on the wall of their project room (see sidebar).[4]

Whenever a user story is stabilized, the developers work through the checklist with the product owner to identify the nonfunctional requirements needed to support the functionality in the user story. Having a checklist visible helps people to be continually conscious of the need for nonfunctional requirements.

SAMPLE CHECKLIST OF NONFUNCTIONAL REQUIREMENTS

10. Look and Feel Requirements
    - 10a. Appearance Requirements
    - 10b. Style Requirements
11. Usability and Humanity Requirements
    - 11a. Ease of Use Requirements
    - 11b. Personalization and Internationalization
    - 11c. Learning Requirements
    - 11d. Understandability and Politeness
    - 11e. Accessibility Requirements
12. Performance Requirements
    - 12a. Speed and Latency Requirements
    - 12b. Safety-Critical Requirements
    - 12c. Precision or Accuracy Requirements
    - 12d. Reliability and Availability Requirements
    - 12e. Robustness or Fault-Tolerance
    - 12f. Capacity Requirements
    - 12g. Scalability or Extensibility Requirements
13. Operational and Environmental
    - 13a. Expected Physical Environment
    - 13b. Requirements for Interfacing with Adjacent Systems
    - 13c. Productization Requirements
14. Maintainability and Support Requirements
    - 14a. Maintenance Requirements
    - 14b. Supportability Requirements
    - 14c. Adaptability Requirements
15. Security Requirements
    - 15a. Access Requirements
    - 15b. Integrity Requirements
    - 15c. Privacy Requirements
    - 15d. Audit Requirements
16. Cultural and Political Requirements
17. Legal Requirements
    - 17a. Compliance Requirements

CONTINUOUS VALUE ANALYSIS

One point of being agile is to continuously deliver value to the business and to be able to respond to change. As discussed earlier, you are more able to respond to change if you know what you are changing from; hence, the value of having a common way of talking about your requirements knowledge. We have looked at the value of having business stories that are the source of user stories. All user stories are traceable back to one or more business stories. Whenever there is a change to the business, the business analyst can identify which business stories are affected and in turn discuss with the developers which user stories are potentially affected. This connection makes it possible to be aware of and plan for the effect of a change.

In addition, we have learned from our clients that a lot of change is brought about by not building the right product in the first place. Effort invested in finding the real business needs is repaid by a higher rate of user acceptance and lower maintenance.

At the start of a piece of work, you can also compare the relative value of the business stories and determine which of them provide the greatest value contribution to the business goal of the project or change. This can and should provide the basis for determining which business story we should sprint for first.

The questions to ask about each business story (and the answers must come from the real business owner) are:

- **Relative to the goal of this project, how much business value would be provided by investing in a new solution to this business story?** Use a scale from 1 (low value) to 5 (high value).
- **Now ask, relative to the goal, how much damage would be done if we do not provide a new solution to this business story.** Once again, use the scale from 1 (doesn't really matter) to 5 (would damage this part of the business).

Table 1 — Business Story Value Analysis

Business Story	Value of Investing (Scale of 1-5)	Damage of Not Investing (Scale of 1-5)
Business Story 1	3	4
Business Story 2	5	1
Business Story 3	5	5
Business Story 4	2	1
Business Story 5	4	2
Business Story 6	5	4
Business Story 7	5	2

This first value analysis, at business story level, identifies which business functions or processes would benefit most from investing in a new solution (see Table 1). The business analyst, as part of his or her daily work, continues to do value analysis at this level to reflect the current state of the business. The intention is for every new piece of development to truly reflect what would be most beneficial to the current state of the business.

The point of the value analysis is to help choose the investment that will provide the highest value given the current realities of the business. It is the business analyst's job to clarify the choices, but making the choice is the responsibility of the business owner. The business analyst acts as a conduit between the business and the developers so that the development teams are aware of what is currently most valuable to the business. This might seem an obvious thing, but recent reports state that many businesses are disappointed by the low level of awareness of the business domain reflected in the software delivered by their agile development teams.[5]

We have suggested that the business story is the best vehicle for communicating the real business. Then, as already discussed, the most valuable business story will spawn multiple user stories. When we have all the user stories for a business story, we can apply the same thinking and thus do a comparative value analysis between those user stories and choose to sprint for the ones that provide the highest value.

INNOVATION

Quite a few clients report that agile is anti-inovation. The developers have a vested interest in developing whatever they can produce within the allowable time. They are rewarded for maintaining the velocity of the project, not for their innovative solutions. Note that innovation, as we use the term here, means fresh thinking. We do not mean that innovation is the same as invention — it's not. Innovation is thinking differently about the business problem with the intention of finding more beneficial things for the business to do.

User stories that are not based on real business stories will struggle to be innovative. The user story describes what happens at the interface and is mostly what the product owner thinks the user wants to do with the software. But without some innovative thinking, it is all too easy to provide just some incremental improvement and let it go at that. Let's look at an example of a user story that was written without any real concern for innovation or the real need that the story is meant to satisfy: "As a bank account holder, I want to check my balance online."

At first, this might seem a reasonable and obvious story. However, it can be made a lot better. Some authors suggest that the "so that" part can be omitted. We suggest very strongly that you always include the reason in your stories. While the Agile Manifesto favors working code over documentation — and there is a lot to be said for that — there is nevertheless a need for the development team to leave behind a guideline of its thinking. Without justifying the requirement — that means including the "so that" — future maintenance teams are deprived of a valuable clue as to why a particular requirement was included in the software product and, hence, what would be the effect of changing it.

Our first question is, "Why does the account owner want to check the balance?" Let's revisit the story and this time look at the reason given for the requirement: "As a bank account holder, I want to check my balance online so that I can access my daily balance 24 hours a day."

This is not exactly a good reason for checking the balance. The "24 hours a day" is slightly more enlightening but doesn't really do more than tell us that the account owner might have nocturnal habits. Why does the account owner wish to check the balance? It is not something we do for fun, so there probably is some business reason behind it. We just don't yet know what that is. Let's make some conjectures.

Suppose the reason for the frequent checking is that the account owner is on a tight budget and is concerned about becoming overdrawn. If so, the owner of the product, presumably a bank, has the opportunity here to be more effective and at the same time provide a better service. Instead of the account owner having to repeatedly check his or her balance to see that it's not going into the red, it would be better to build a feature that notifies the account owner if the normal monthly payments such as rent, electricity, school fees, and so on, will reduce the account balance to zero or beyond: "As a bank account holder I want to be informed if my monthly balance is projected to go to zero or below so that I can arrange for an overdraft."

Furthermore, we suggest that it is far more useful for the account owner to be periodically informed of the amount of discretionary money in the account: "As a bank account holder I want to be informed of my projected balance after all regular monthly payments have been deducted so that I know how much I can safely spend."

Having a feature that lets the account owner check the balance online is the simplest feature to have. However, we suspect that without any kind of business thinking, or innovative thinking, it would probably turn out to be a feature that did not solve the real problem. If other banks solve the real problem (i.e., they understand the real needs of their customers and offer this service to attract more customers), then the original story could hardly be said to providing real business value.

When a business analyst investigates a business story, he or she is consciously trying to encourage innovation by first making abstractions that uncover the real business problem. Figure 7 illustrates the skill of looking at the same business story from several different points of view and thereby discovering undreamed of innovations that could make a significant difference to the business.

When a business analyst investigates a business story, he or she is consciously trying to encourage innovation by first making abstractions that uncover the real business problem.

In the bottom-left quadrant of the Brown Cow model[6] the viewpoint focuses on *how* things are done *now*. This is commonly what business people ask for — a little bit more of what we already have. In the bottom-right quadrant the viewpoint focuses on *how* things could work in the *future*. Once again, it is likely that someone who has a requirement will express it in these terms. In other words, like our nurses in the previous example, rather than asking for what they really need, they ask for a solution.

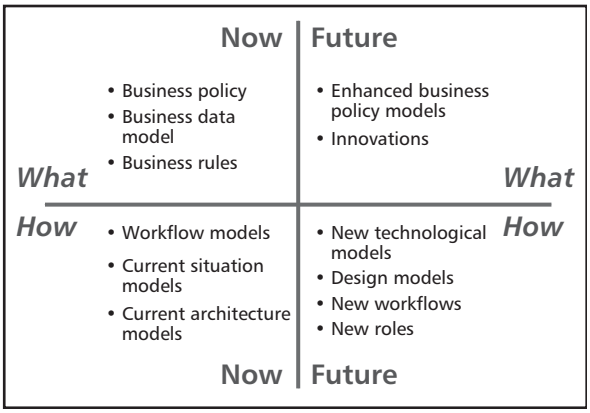


Figure 7 — The Brown Cow model illustrates four points of view that help to uncover the real business problem and identify useful innovations.

As part of his or her analytical skills, the business analyst also learns to explore the problem in a solution-neutral fashion. We refer to this as the ability to "think above the line." The line in this case is the horizontal line that separates the top and bottom quadrants of the model — the abstract thinking about the real problem from the technological view of the solutions.

The top-left quadrant focuses on *what* we do *now*, independent from how it is done now or how it might be done in the future. This view uncovers the essence of the problem: the business rules and the business data that has to be there independent of any solution. Above the line, the business analyst exposes the real business requirements by stripping away all solution-oriented aspects and thereby coming up with a policy-only statement of what the business is really doing. The top-right quadrant of the model is where the business innovation happens. It is here that the business analyst can make suggestions about improving business rules or using existing business data to be able to make better business decisions.

Innovative business value discovered by this sort of innovative thinking are then included in the business stories. These are innovations that could never be discovered by focusing on a software interface.

MAXIMIZING BUSINESS VALUE

As discussed in this *Executive Report*, we are concerned that agile techniques, or the way that some organizations are implementing them, are potentially shortchanging the business. Our concern is that the connection between the development team and the real business is not strong enough. The role of the product owner, unless it is held by someone with exceptional training or experience, is not sufficient to provide the necessary business specific input to the developers. To be effective, the connection to the business requires multiple viewpoints, multiple skills, and, unfortunately, multiple people. One person is far too little bandwidth to make an effective link.

User stories can be a significant part of the problem. Too often, user stories describe a solution to an (often) unstated problem, and are part of our concern. By leaping directly into a solution, there is a danger that the real need is overlooked in favor of speedy development. However, no amount of speed or efficiency can make up for the lack of meeting the owner's needs. If they are to be effective, user stories must provide an insight into the real need. This gives the developers the best chance of delivering a genuinely useful and valuable product.

Business value can only be maximized when the delivered product is optimally beneficial to the owner. The real owner — not the product owner — is the one who must be given priority when the product is developed. When the user stories are derived from well-analyzed business stories, then we can better connect agile development with the owner's work. This means that we need to consider business analysis and systemic thinking to be a normal part of the agile development process. And, of course, innovation must also be thought of as a normal development activity.

These things are possible to do without altering the core values of agile development. In fact, when we think of customer collaboration, it is important that we think of giving the customer a better way of communicating and not forcing the conversation to be about solutions, but about needs. When we can understand the real needs, and respond to them, guide them, and possibly improve them through innovation, then, and only then, can we supply the maximum business value to our clients.

ENDNOTES

[1]Seddon, John. *Freedom from Command & Control: A Better Way to Make the Work Work.* Vanguard Consulting Ltd., 2003.

[2]For more on the requirements knowledge model, see: Robertson, Suzanne. "Requirements for Managing Requirements." Cutter Consortium Agile Product & Project Management *Executive Report*, Vol 8, No 9, 2007.

[3]Cooper, Alan, Robert Reimann, and David Cronin. *About Face 3: The Essentials of Interaction Design.* Wiley, 2007.

[4]Taken from the Volere Requirements Specification Template (www.volere.co.uk/templates.htm).

[5]Conboy, Kieran, Sharon Coyle, Xiaofeng Wang, and Minna Pikkarainen. "People Over Process: Key Challenges in Agile Development." I*EEE Software*, July/August 2011.

[6]The Brown Cow model is so called because it reminded some business analysts of the English elocution lesson "How Now Brown Cow" (www.volere.co.uk/pdf%20files/howNowBrownCow.pdf).

RECOMMENDED READING

Cockburn, Alistair. *Agile Software Development: The Cooperative Game*. Addison-Wesley Professional, 2006.

Cohn, Mike. *User Stories Applied: For Agile Software Development*. Addison-Wesley Professional, 2004.

Leffingwell, Dean. *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise (Agile Software Development Series)*. Addison-Wesley Professional, 2011.

Robertson, James, and Suzanne Robertson. *Mastering the Requirements Process*. Addison-Wesley Professional, 2006.

Senge, Peter. *The Fifth Discipline*. Crown Business, 2006.

Tockey, Steve. *Return on Software: Maximizing the Return on Your Software Investment*. Addison-Wesley Professional, 2004.

ABOUT THE AUTHORS

James Robertson is a Senior Consultant with Cutter's *Agile Product & Project Management* practice. As a principal and founder of the Atlantic Systems Guild, he is known for his work in implementing systems engineering principles that link business specialists with solution specialists. Mr. Robertson is a consultant, lecturer, author, and project leader who focuses on the requirements for software as well as the contribution that good requirements make toward successful projects. His training as an architect has led to his work on good design principles and his focus on how to integrate innovation into the jobs of requirements specialists. Current work includes developing heuristics for applying abstraction in order to understand and communicate the real business problem as distinct from the solution. Mr. Robertson is the coauthor of five books, including *Mastering the Requirements Process* and *Requirements-Led Project Management*, and co-inventor of the Volere requirements techniques. His work with his colleagues in the Atlantic Systems Guild led to the Jolt awarding–winning book *Adrenaline Junkies and Template Zombies: Project Behaviour Patterns*, which recognizes and defines patterns of project behavior. He can be reached at jrobertson@cutter.com.

Suzanne Robertson is a Senior Consultant with Cutter's *Agile Product & Project Management* practice. As a principal and founder of the Atlantic Systems Guild, she works with organizations in applying innovative techniques and fresh thinking to systems development activities. Ms. Robertson is coauthor of *Requirements-Led Project Management*, a book that provides project managers with guidance on how to use the work done by requirements analysts as input to steering projects. This -management book connects to her coauthored book *Mastering the Requirements Process*, a guide for practitioners on finding requirements and writing them so that all stakeholders can understand them. She is also coauthor of the Jolt awarding–winning book *Adrenaline Junkies and Template Zombies: Project Behaviour Patterns.* Her current work includes research and consulting on the management, sociological, and technological aspects of requirements. The product of this research is Volere, a complete requirements process and template for assessing requirements quality and for specifying requirements. Ms. Robertson is author of many papers on systems engineering. She also speaks at numerous conferences and universities. She is a member of IEEE and on the board of the British Computer Society's Requirements Groups. Ms. Robertson was the founding editor of the Requirements Column in *IEEE Software*. She can be reached at srobertson@cutter.com.

Israel Gat, Practice Director

CONSULTING

Technical Debt Assessment and Valuation

A Universal Tool for Evaluating, Governing, and Managing Software Projects

Do You Need a Technical Debt Assessment?

Are you a CIO looking to ensure delivery over development?

A CTO in search of early warning signs your development project is in trouble?

An M&A/due diligence investigator in need of assurance that the code you're acquiring isn't toxic?

A CEO responsible for governing the development process effectively and ensuring the execution of corresponding go-to-market plans in a reliable manner?

A venture capitalist determining how much (more) money to invest in your portfolio company?

Cutter's Technical Debt Assessment and Valuation is customized to meet your specific needs. For details, contact your Cutter Account Executive at sales@cutter.com or +1 781 648 8700.

Technical debt is a real cost. Whether you're looking at it from the perspective of a venture capitalist or CEO, or from the viewpoint of a CIO or CTO, or you're trying to determine if a merger or acquisition makes sense, knowing how much money is required to "pay back" your software's technical debt may be the very factor that proves your decision to be a good one or a very costly one.

In a Technical Debt Assessment and Valuation, Cutter's Senior Consultants — led by Practice Director Israel Gat — examine the quality of the software under examination through technical and business lenses. Whether that code is your own, has been developed by an acquisition candidate, or by a company you're investing (more) in, Cutter's Technical Debt Assessment and Valuation will enable you to:

- Get the vital answer to the question, "Is your software an asset or a liability?"
- Know how much (more) money you will need to invest in order to fix the code
- Get data and insights you need to guide the fix-it process for the software
- Identify projects that are likely to get in trouble at an early stage of the software lifecycle
- Determine if the technical debt is keeping your software development staff from responding quickly and effectively to customer requests

Plus, you'll get the tools you need to govern the software development process on an ongoing basis to avoid the expense of future technical debt.

Do you know the true value of your software? Are you sure? Does your value calculation include technical debt?

In a Technical Debt Assessment and Valuation, Cutter's Senior Consultants will identify the architecture, design, coding, testing, and documentation deficits that constitute technical debt. The assessment combines static code analytics with dynamic program analytics to give you "x-rays" of the software being examined at any desired granularity. You'll get a report and/or presentation that provide you with a dollar figure you can plug into your financial models so that you can objectively analyze your critical software assets. Easy-to-understand graphics depicting the quality of your code and the cost of your technical debt will enable your team to zero in on the most hazardous projects and fix them in a prioritized manner. And you'll get operational recommendations that take into

Agile Product & Project Management

account various qualitative and quantitative factors that characterize your software development process. These recommendations will help you make the best decisions about your ongoing strategy for this software development effort.

Cutter's Technical Debt Assessment and Valuation is most effective as an on-premises engagement. However, it can also be done as a largely off-premises engagement based on a snapshot of the code. For more details, or to arrange your Technical Debt Assessment and Valuation, contact your Cutter Account Executive at +1 781 648 8700 or sales@cutter.com.

"We'll have a LOT of crappy systems to fix up 5-10 years from now."

— Ed Yourdon, Cutter Fellow

"Death March Projects in Today's Hard Times,"
Boston SPIN, 16 March 2010

Cutter Research & Opinion on Technical Debt

"Technical Debt" from *Cutter IT Journal*
— Israel Gat, Guest Editor
(www.cutter.com/itjournal/fulltext/2010/10/index.html)

"Delving into Technical Debt"
— Chris Sterling and Israel Gat
(www.cutter.com/content/project/fulltext/updates/2011/apmu1120.html)

"Servicing Technical Debt"
— Israel Gat et al.
(www.cutter.com/content/project/fulltext/advisor/2011/apm110623.html)

"Quantifying the Start Afresh Option"
— Israel Gat
(http://blog.cutter.com)

"Enterprise Architecture, Technical Debt, and Technical Paralysis"
— Ken Orr
(www.cutter.com/content/architecture/fulltext/advisor/2011/ea110817.html)

"To Release No More or To 'Release' Always: Part II — Toward a New Business Design for Software"
— Israel Gat
(www.cutter.com/project/fulltext/updates/2008/apmu0823.html)

"The Agile Triangle — Quality Today and Tomorrow"
— Jim Highsmith
(www.cutter.com/project/fulltext/advisor/2010/apm100401.html)

Agile Assessment

Cutter's Technical Debt Assessment and Valuation is extremely synergetic with our Agile Assessment, a quantitative and qualitative analysis of an organization's use of Agile methods, its software engineering practices, and its project management skills and capabilities. When the two are conducted jointly, Cutter will present your team with a composite plan for fixing software quality deficits and software process deficits in tandem.

Cutter Consortium
37 Broadway, Suite 1, Arlington, MA 02474-5552, USA
Tel: +1 781 648 8700; Fax: +1 781 648 8707; www.cutter.com; sales@cutter.com

About the Practice

Agile Product & Project Management

From applying the "nuts and bolts" of Agile, to transforming your organization at the enterprise level, to applying technical debt to reduce risk and improving software governance, Cutter offers the consulting, training, and research you need to make your Agile initiative a source of strategic competitive advantage.

Cutter and its experts are pioneers and leaders in the Agile movement. For more than a decade, our team, lead by Dr. Israel Gat, has helped organizations around the world transition to and/or solidify their project management and software development techniques, Agile engineering practices, and software governance. With a tight focus on Agile principles and traits — delivering customer value, embracing change, reflection, adaptation, and so forth — Cutter has successfully helped many teams both shorten product development schedules and increase the quality of the resulting products.

Your team will find many forums in which it can interact with Cutter's Agile experts to brainstorm, debate, and learn. From research and inquiry privileges to regular virtual meetings with Cutter's Agile thought leaders, participation in webinars, and Agile-specific Q&A sessions, the opportunities to discover new strategies, gain insight into how to launch new practices, secure support for your governance strategies, and more are boundless.

Products and Services Available from the Agile Product & Project Management Practice

- The Agile Membership
- Research & Analysis
- Inquiry Response
- Consulting
- Inhouse Training & Executive Education
- Mentoring
- Research Reports

Cutter Consortium Practices

Each Cutter Consortium practice includes a subscription-based research service, plus consulting, training, and executive education services:

- Agile Product & Project Management
- Business & Enterprise Architecture
- Business Technology Strategies
- Data Insight & Social BI

Senior Consultant Team

The Cutter Consortium Agile Product & Project Management Senior Consultant Team includes many of the trailblazers in the project management/peopleware field, from those who've written the textbooks that continue to crystallize the issues of hiring, retaining, and motivating software professionals, to those who've developed today's hottest Agile methodologies. You get sound advice and cutting-edge tips, as well as case studies and data analysis from best-in-class experts. This brain trust includes:

- Israel Gat, Practice Director
- Scott W. Ambler
- Jurgen Appelo
- Christopher M. Avery
- Brent Barton
- Sam Bayer
- Kent Beck
- E.M. Bennatan
- Tom Bragg
- Robert N. Charette
- Alistair Cockburn
- Jens Coldewey
- Ken Collier
- Ward Cunningham
- Patrick Debois
- Tom DeMarco
- Esther Derby
- Khaled El Emam
- Hillel Glazer
- Ron Jeffries
- Mark Levison
- Tim Lister
- Alan MacCormack
- Masa K. Maeda
- Michael Mah
- Ken Orr
- James Robertson
- Suzanne Robertson
- Alexandre Rodrigues
- Dave Rooney
- Johanna Rothman
- Andrew Shafer
- Hubert Smits
- David Spann
- Chris Sterling
- James Sutton
- Rob Thomsett
- Jim Watson
- Robert K. Wysocki