



## Can Agile Work for This Project?

### Executive Summary

One of the four pillars of the “Manifesto for Agile Software Development” is that “we have come to value... working software over comprehensive documentation.” In reality, we are saying that we value working systems as opposed to merely limiting ourselves to only delivering working software. Most of the very large and complex projects we deliver are really as much about integrating systems together as they are about building new bespoke software applications. But do all of these different flavors of projects lend themselves towards using Agile? This is a question that is on the minds of many decision makers in the field of information technology these days.

Many organizations have already adopted Agile as the primary software development methodology for custom building new applications. However, many of the projects that are undertaken by information technology organizations don't fit neatly into the paradigm of a brand-new application. IT projects come in many sizes, shapes and flavors. Based on our experience virtually all of them can be successfully done using Agile, but we need to be smart about it and make sure we carefully think through how we would adopt Agile for these different patterns of project types.

Before we can say definitively that a given project can be done in Agile, we need to look at answering four important questions. If we are able to answer these four questions with well-thought-out responses then we would be fairly assured we could have success with Agile for a

given project. If we find that we are struggling with coming up with answers for these questions, then we probably need to spend a little more time thinking through how we would apply Agile before we move forward with the decision to use it on that given project.

### What Do Our User Stories Look Like?

The first question we should be asking is: What is our concept of a user story? Are we describing user interactions with a system; are we describing a feature; or are we describing a component? What other types of artifacts will we need to produce to augment the user story? At the same time, we also need to keep in mind the Agile Manifesto, and remind ourselves that we value working systems over comprehensive documentation. So we don't want to go overboard, but instead find that balance of how much documentation is the “sweet spot” for building working systems.

### How Will We Sequence Our Backlog?

The second question is: What criteria are we going to use to determine the order of user stories on the product backlog? Can we attach some sort of monetary value to the user story and thereby order the more valuable stories to the top and the less valuable to the bottom? Should we be prioritizing the stories based on architectural uncertainty or complexity, or whether they are foundational stories that later stories on the backlog are inherently dependent upon? Or in the example of a project where the solution is more about standing up a new commercial off-the-shelf

(COTS) application, perhaps what goes onto the backlog is a sequenced set of configuration tasks that need to be done in a particular order.

### How Will We Partition Tasks?

The third question we should be asking is: Can we partition the user stories into sets of tasks that the Scrum team can perform within a Sprint to deliver some sort of “potentially shippable” product that can be demonstrated to the product owner? Perhaps these tasks are the typical set of tasks you would expect to see in a traditional project where you are building a new application: perform design, coding and unit test, build and integration. Perhaps the tasks are a different set of tasks due to the nature of the project we are performing, such as configuration tasks, data reengineering activities such as extract, transform and load or perhaps they are tasks that would be performed while using a 3GL or CASE tool to generate code.

### What Is Our Definition of Done?

The fourth and final question is: How will we know we are done with the user story? That is to say, how do we know when we have something that is “potentially shippable”? How are we going to test what we are delivering? Are we going to use test-driven development (TDD)? Are we going to have a separate bifurcated testing team downstream of the Scrum team to certify the product coming out of the Sprints before it is actually shipped to customers? These are important questions to consider when deciding how we are going to test our stories.

For any given IT project, if we are able to come up with a well-reasoned set of answers to all four of these major questions, then we can feel reasonably assured that choosing Agile for that specific project was a good decision and that there is not a particular nuance that prohibits the use of Agile. Now let’s reinforce this by looking at some common examples of archetype patterns of information technology projects that probably look pretty similar to some projects on your road map or portfolios. Then we’ll discuss how we would answer these four questions for each of those patterns.

### Custom-built/Bespoke

The green field project where we are building a brand new bespoke application from scratch is the most classic example of a project well suited for Agile. Whether the skill set is Java, .NET or a legacy mainframe technology is almost

immaterial when we are trying to answer the four big questions. Moreover, the way in which a given organization would answer those questions will probably have the least variability from organization to organization in this example. For the sake of this example, let’s assume we are talking about a software company that makes online banking software using a fairly traditional implementation of Scrum.

User stories will most likely look like the classic 3x5 card that describes who the actor is, what they are trying to accomplish and how we will know if we have successfully accomplished it. It would resemble something like, “As a customer, I want to be able to see my last thirty days of transactions on my account so that I can reconcile my checkbook.”

The set of user stories would most likely be ordered numerically with an absolute ranking on the product backlog by the product manager. It would then be the product manager’s responsibility to come up with the rationale for how the ranking is performed. In some organizations, some sort of monetary value would be placed on the user story (for example, some user stories might be related to a new product feature that would allow us to sell our product to new customers and create incremental revenue). Another possibility is that the product manager might rank stories based on how many end customers have requested a given feature or capability.

Partitioning the story into tasks is also a pretty straightforward exercise, wherein there might be some tasks oriented around creating technical stories. These would be performed in the early part of the iteration/Sprint, then there would be coding and unit testing tasks, and then finally some acceptance and validation tasks.

Lastly, it would most likely be a responsibility of the product manager to answer the question, “are we done?” The user stories would likely have some acceptance criteria that can be used to make that determination, and when the product manager accepts the story and it is shown in the demo then the story would be considered “potentially shippable.” It would then ship during the next scheduled release.

### Package Implementation/ERP

Package implementations and enterprise resource planning (ERP) are other common patterns of IT projects. But how would we go about answering the four big questions in this type of example?

Let's assume we are talking about an organization that is migrating from a set of home-grown order management and fulfillment applications to a commercial off-the-shelf (COTS) package. It's likely that most of the user stories would resemble the types of user stories from the first example; they would describe interactions between end users and the system, explain what business function they are trying to perform and describe some sort of acceptance criteria that tell us when that functionality or capability has been provided.

However, the role of the product manager and the way in which user stories are sequenced on the backlog are both likely to be different in this example. Instead of some sort of ranking based on value or desirability of a feature, it is more likely that there are certain core modules within the target package that have to be stood up and configured first before other parts of the ERP package can work. For example, you need to set up and load the product module before you can start setting up and configuring the order entry applications, so the stories might get ordered based on these dependencies and would look something like a critical path schedule.

The tasks that the delivery team would perform would also be different from the traditional custom application development tasks of design, coding and unit testing. A lot of the technical work performed is more oriented around configuration as opposed to software development. But all that means is that we would partition the story into a different set of tasks that are more in tune with the type of technical work that would be performed. Perhaps the story would be partitioned into a set of tasks such as "configure hierarchies," "customize business rules" and "validate functionality in the user interface."

It is likely that the acceptance criteria on the stories will be similar to acceptance criteria on user stories for projects where we are custom building. The stories would describe a working functionality where an end user is able to perform some important business-oriented task such as entering an order. The acceptance criteria would explain some sort of evaluation criteria that would describe how we would know when we are successful. Therefore the product manager could play a similar role of formally accepting the stories and then having them demonstrated at the end of a Sprint. So, much like in the bespoke example, the product manager would be the gatekeeper determining which parts of the overall package

implementation have been delivered against the acceptance criteria and are now "potentially shippable."

## Data Warehouse/Business Intelligence

Data Warehousing and BI projects are an interesting example. These projects tend to be complex, very large and long in duration. They also tend to be very discovery driven especially as it relates to data cleansing and extract, transform and load (ETL). In addition to having a considerable amount of effort relating to ETL, there is usually some sort of analytical tool or reporting package that is implemented. In general, we could treat the standing up and configuring of the reporting package as being analogous to the package/ERP project pattern which we have already discussed. So let's focus the discussion on how to do data conversion and data migration projects using Agile.

ETL projects can be done using a variety of technologies and tools, some involving custom coding, legacy and more modern programming languages and ETL tools. What they have in common, though, is moving and synchronizing data from a source to a target, and performing transformations on the data. Sometimes we are going from source systems to an operational data store (ODS). Sometimes we are loading large data warehouses, and sometimes we are extracting from data warehouses and loading data marts.

User stories in this example would probably look dramatically different than user stories from a custom application development project. We are less likely to be describing the way an actor interacts with a software application. Rather, we are likely describing an extraction routine that needs to pull data from a source system, or describing business rules that need to be applied to data to transform it into the target schema.

We may need to give some careful thought on who is going to play the role of the product manager and how that person would prioritize the stories on the backlog for the development team. One approach could be that the data architect who designed the target schema could play that role. Also, the sequencing of the backlog could be based on the chain of inheritance in the data model. That is to say, the early stories on the backlog might be related to extracting and loading the primary keys and identifying data of base stand-alone entities such as customer, location and product. Then the middle priority stories might end up being about ETL routines dealing with the one-to-many and

many-to-many relationships like sale and sale item. And then the latest stories might be about populating important but ancillary data elements on the tables in the data warehouse.

Also, if the project is about both loading a data warehouse and data marts, the early sets of stories would likely be about going from source systems to the warehouse and the later stories might be about populating downstream data marts used by analytical tools.

Partitioning user stories into development tasks will also be a challenge. Many times in these types of projects the work that is done is very discovery driven. Frequently, the quality of the data itself is an unknown. One approach to try would be to have one story about doing the 80% of the customer records and then having developers use ETL tools in an early Sprint to do the 80% accepting the fact that the first time through the gate we might have a considerable amount of data that is fallout because of unexpected data issues. Then in a second trailing Sprint the developers would tackle the 20% that required separate special treatment because of more complex business rules that would need to be identified in other user stories.

Answering the questions about when we are done is also a challenge. One way to look at it would be to have two levels of “doneness.” One early grade when the data in the warehouse is now usable and complete enough that we can make good business decisions based on how the data is flowing into the data marts. Perhaps after 80% of the sales data is flowing cleanly we can use that incomplete set to still make pretty reliable product marketing decisions. Then when the data quality hits the higher grade later on we can strive closer towards 99% to 100% quality. This may be a challenge based on what types of business decisions are being made with the information coming out of the warehouse and data marts.

## Footnote

<sup>1</sup> <http://agilemanifesto.org/>

## About the Author

*Dan Fuller is an Associate Director in Cognizant's Advanced Solutions Group. He is an experienced Agile practitioner who has led some of Cognizant's largest and most complex deliveries for clients using Agile. His background includes 20 years of information technology consulting for some of the largest and most respected IT consulting firms. He received a master's degree in technology in education from Harvard University and a bachelor's degree in accounting and information technology from the University of Massachusetts, Amherst. He can be reached at [Dan.Fuller@cognizant.com](mailto:Dan.Fuller@cognizant.com).*

## Application Value Maintenance

Application value maintenance (AVM) projects are usually ongoing and about making minor enhancements and making rapid break-fixes to a mature application on the right hand side of the product lifecycle.

The user stories are likely to be very simple, sometimes being as straightforward as a defect or customer issue logged in a quality tool.

Some sort of algorithm based on priority and severity is probably going to be used to determine the sequence of the issues on the defect backlog. Sometimes when it's about customer issues we could use a first in first out (FIFO) approach to sequencing. Or perhaps depending on the particular customer there are different service level agreements (SLA) that dictate the priority.

Because time is usually of the essence, we may want to have the partitioning of stories be as simple as possible – basically, just get it done. Whatever developer is now free takes the next item off the backlog, works it to completion, marks it ready for test and then immediately moves to the next item. No Sprints, no demos. In fact, this sounds a bit like Kanban, as lean as we can be.

Then it would usually be up to some quality organization to make the call on whether the issue has been resolved and can be moved to a closed status, or whether the issue persists and needs to go back to the top of the backlog.

## Deciding on Agile

Coming up with thoughtful well reasoned answers to these questions is certainly not a guarantee that we will have success with Agile. There are many other dimensions we need to assess, such as learning and maturity, organizational cultural fit, availability of resources, etc.



---

## About Cognizant

Cognizant (NASDAQ: CTSH) is a leading provider of information technology, consulting, and business process outsourcing services, dedicated to helping the world's leading companies build stronger businesses. Headquartered in Teaneck, New Jersey (U.S.), Cognizant combines a passion for client satisfaction, technology innovation, deep industry and business process expertise, and a global, collaborative workforce that embodies the future of work. With over 50 delivery centers worldwide and approximately 130,000 employees as of September 30, 2011, Cognizant is a member of the NASDAQ-100, the S&P 500, the Forbes Global 2000, and the Fortune 500 and is ranked among the top performing and fastest growing companies in the world. Visit us online at [www.cognizant.com](http://www.cognizant.com) or follow us on Twitter: Cognizant.



**World Headquarters**  
500 Frank W. Burr Blvd.  
Teaneck, NJ 07666 USA  
Phone: +1 201 801 0233  
Fax: +1 201 801 0243  
Toll Free: +1 888 937 3277  
Email: [inquiry@cognizant.com](mailto:inquiry@cognizant.com)

**European Headquarters**  
1 Kingdom Street  
Paddington Central  
London W2 6BD  
Phone: +44 (0) 20 7297 7600  
Fax: +44 (0) 20 7121 0102  
Email: [infouk@cognizant.com](mailto:infouk@cognizant.com)

**India Operations Headquarters**  
#5/535, Old Mahabalipuram Road  
Okkiyam Pettai, Thoraipakkam  
Chennai, 600 096 India  
Phone: +91 (0) 44 4209 6000  
Fax: +91 (0) 44 4209 6060  
Email: [inquiryindia@cognizant.com](mailto:inquiryindia@cognizant.com)