

Project Expectations: The Boundaries for Agile Development

Diana Mekelburg
Extreme Project Management

How can you manage agile development when its practitioners value “working software over comprehensive documentation” and “responding to change over following a plan”? Control the boundaries. Manage the project expectations instead of waiting for requirements and plans to miraculously stabilize.

Isabella, a software development executive, listened quietly while Ravi, the project lead, reported what terrific success the project team was having with the eXtreme Programming (XP) methodology. Sitting next to him, the project sponsor smiled and nodded. Months before, when the software group had proposed using agile software development on the project, Isabella had approved their request without realizing that they intended to use something with *extreme* in its name. She also had not realized that the only requirements and planning documents that she would see would be very high-level – too sketchy in her opinion to control the project.

When Ravi finished regaling the meeting participants with stories of the project’s wonderful atmosphere of camaraderie, Isabella asked whether the system would be ready to install by the deadline. Ravi said that he could see no reason why it would not and flipped forward to the slide of his recently revised high-level schedule. Isabella felt her stomach churn as she searched the schedule for deliverables and found only activities. Terms like *refactor*, *user stories*, and *iteration* made her wonder whether they were building software at all or, like their earlier experiments in object-oriented design, were endlessly refining object components that no one would ever use.

She asked Mary, the project sponsor, what she thought of the project. Mary replied that she was very pleased with the demos that she had seen so far. When Isabella asked her if she thought the project was on schedule, Mary turned to Ravi for the answer. Isabella had heard enough. After the meeting had ended, she told Ravi privately that from now until the end of the project she would meet with him weekly, instead of monthly, to review the project’s status. She ordered him to bring complete requirements specifications and a detailed project plan to the next meeting for her review. After some heated dis-

cussion, Ravi stormed out of the room.

Later, Isabella told one of the other executives, “You would think that after all the money we have spent on training this organization in project management, at least a few of them would understand the concept of project scope.”

Agile Scope Management

Project scope management can take on a whole new philosophy and appearance with agile software development. This can be a formidable challenge for tradi-

“Traditional scope definition has always been a thin security blanket that cannot protect software projects from the crashing waves of scope change in volatile projects.”

tionally trained managers like Isabella. Although the intent of agile software development is to produce the best product possible in the least amount of time and for the least amount of cost, the result is often scope management that appears to be more improvisation than controlled execution. In these cases, the traditional decomposition approach to project planning and prediction is not possible.

In traditional software development methodologies, product scope is typically defined in a top-down manner, starting with high-level requirements that are decomposed to more specific requirements. The project manager can use a

parallel approach for defining project scope by building a work breakdown structure. This approach gives management a progressively more accurate estimate of the time and cost to complete the project, i.e., as the product and work are specified in greater detail, the project estimate becomes more accurate. Once the project’s scope baseline has been set, software managers’ main concern for managing scope is to guard constantly against scope creep, especially in the form of product feature changes.

These methods work well when the product definition is not too complex, controversial, or volatile. However, in many cases, the product is excessively difficult to define, and these methods are unreliable, misleading, and conflict-ridden. It is no wonder that software developers are willing to adopt a lighter, potentially more effective approach, such as agile software development. But where does this leave scope management?

The Agile Software Development Manifesto values working software over comprehensive documentation and responding to change over following a plan [1]. As Alistair Cockburn explains, requirements can be imperfect, and design documents and project plans can be out of date, yet the project can still succeed by applying such principles as communication and community. This can leave the traditional software project manager adrift in a sea of change, clinging to a frail life raft lashed together from in-person visits, whiteboard sketches, invention, and light-and-sloppy methods [2]. As scary as this image is, it is not new. Gause and Weinberg explored the notion that requirements documents are less important than the process of defining them back in 1989 [3].

Traditional scope definition has always been a thin security blanket that cannot protect software projects from the crashing waves of scope change in volatile projects. Even under contract, software scope is subject to disputes and threats of litigation. Traditional scope documents such as requirements defini-

tions and project plans do give software management a starting point for negotiating scope changes. Without these, the manager of an agile software development project seems to have little to bring to the negotiation table.

Outcome Expectations and Scope Management

The flaw in both traditional and agile approaches is the assumption that project success is determined by delivering specific product features, whether they are defined hierarchically through decomposition and change management or through collaborative iterations. Software development projects are successful only when they have met the stakeholders' expectations, the most important of which are not limited to specific product features.

There are three classes of stakeholder expectations: business outcome, project conduct, and product. Software development invariably focuses on product expectations as early in the project as possible. Meeting specific product expectations is more predictable, manageable, and, for software builders, more enjoyable than trying to meet either business outcome or project conduct expectations. However, this focus on the product exacerbates scope management problems, especially in iterative development such as XP. Repeated efforts to get the product functionality just right can lead to extraneous functionality or more elaborate functionality that works against stakeholders' other expectations. The key to controlling scope in iterative development is committing to business outcome expectations as the outer boundaries of scope.

Business outcome expectations are the effects that stakeholders expect the software development project to have on internal operations and/or on marketplace or other external environments. An example of an operational outcome expectation is reducing inventory errors in manufacturing. An example of an external outcome is capturing a segment of the personal digital assistant (PDA) market. Outcome expectations may align with corporate strategic goals, depending on the clarity, viability, or influence of the strategic goals.

Delivering the right software product features depends ultimately on whether those features support the business outcome expectations. This is especially important in agile projects. Without stable requirements to bolster or burden them, agile software development proj-

ects need clearly defined and committed business outcome expectations to contain them.

A key control device in XP, for example, is the story. Each increment of the product is planned to implement a story that represents a set of user functionality. The project sponsor, other users, and the development team decide jointly which story is to be implemented next, and how it will be implemented in product features. The sequence of stories can wander far from the original intentions for the project. Similarly, the features chosen to implement each story are defined iteratively and can also wander. In cases where either the development team or the customer is committed to some limited budget or timeframe for the overall project, this wandering can lead to problems in funding and deadlines.

XP and many other types of agile projects are expected to shift direction.

“The project team, which includes the project sponsor, commits to selected business outcome expectations early in the project and repeats this process throughout the project as major changes in the business environment occur.”

Requirements volatility is a primary reason for selecting extreme methodology. However, changes can be contained. If the project team has collected, validated, and committed to meet a set of compatible and feasible business outcome expectations, they can use them to open negotiations about which shifts in direction to apply. They compare each iteration plan and/or user story to the committed expectations. Product functionality that is in line with the overall committed expectations are changed routinely, while changes that contradict or modify the committed expectations are handled as major changes in project scope.

The project team, which includes the project sponsor, commits to selected business outcome expectations early in the project and repeats this process throughout the project as major changes in the business environment occur. The wording for a business outcome expectation is, “As a result of this project, [some group] will be able to [do something].” This simple statement is supported by a collection of justifications, criteria, and evaluations. Outcome expectations do not refer to a specific product feature, nor do they define specific business functions. This allows the project team leeway in selecting the best detail solutions to meet the expectations.

Not all expectations can be met. Many compete for resources, a few directly contradict each other, and some are not justifiable. The most challenging step in committing to the project's outcome expectations is selecting the expectations to be met. Expectations that are rejected or deferred introduce risks to the project in terms of distractions and competition. The primary objective of commitment is a consistent vision of the project's success in terms of the expectations that the project is committed to meet. A secondary but crucial objective is to develop a plan for mitigating or responding to the risks of deciding not to meet other expectations.

Product Expectations and Outcome Expectations

The relationship between software products and business outcomes is often complex or tenuous, depending on a number of organizational factors. Few sponsors or users can envision the bridge between technology and big-picture business outcome, no matter how well they learn to match technology to immediate functionality. As the manager of an agile project, you can either evaluate these product-outcome relationships overtly early in the project, or you can rely on users and the project sponsor to keep the project on track by guessing.

To control project scope, every product expectation must support a committed business outcome expectation. In any product-centric methodology such as XP, every story and every product feature must be checked against the committed business outcome expectations. When they do not match, either the product feature (or business function) is out of scope and irrelevant, or the committed expectation is out of sync with reality.

Conduct Expectations

Conduct expectations are what the stakeholders expect to experience as part of the project. For agile development, the participation of the project sponsor, for example, can be extremely important. However, if the project sponsor expects to take a hands-off or infrequent visitor approach to overseeing the project, methodologies such as XP cannot be used effectively.

Conduct expectations include executives' perception of how predictable the completion of the project will be, and how much control and/or documentation the project will produce. In Isabella's story at the beginning of this article, she expects more documentation and supporting data for the project manager's predictions. If the project team had identified her expectations about the project's conduct as well as the project sponsor's and the project team's expectations, Ravi would not have been taken by surprise. Isabella would have had a chance to negotiate some compromise reporting.

As with product expectations, project conduct expectations must support the committed business outcome expectations. For example, if Isabella's concern for meeting the deadline is tied to a crucial business outcome, then expanding the

usual project management activities for an XP project could have been justifiable.

Conclusion

Agile software development challenges software managers and sponsors to give up their reliance on comprehensive documentation and intermediate work products. However, without boundaries, the iterative definition and development of product functionality can range out of control. By committing to compatible and feasible business outcome expectations, development teams can manage the scope

of agile projects successfully – to the satisfaction of sponsors, executives, and users. ♦

References

1. Cockburn, Alistair. Agile Software Development. Addison-Wesley, 2002: 213.
2. Cockburn, Alistair. Agile Software Development. Addison-Wesley, 2002: 177.
3. Gause, Donald, and Gerald Weinberg. Exploring Requirements. Dorset House, 1989: xvi.

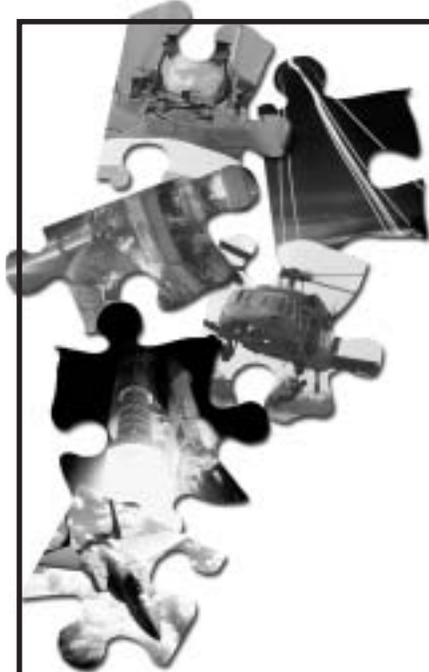
About the Author



Diana Mekelburg, Project Management Professional and Certified Software Quality Engineer, is an information systems management consultant, trainer, and coach. She has taught project/program management classes to more than 500 people. Mekelburg has managed software development in a variety of organizations, including information technology, contract,

and commercial during three decades of software development, from mainframe to e-commerce. She has participated as an assessor on two large-scale Software Engineering Institute Capability Maturity Model® (CMM®) assessments and has taught CMM classes to more than 200 people.

Extreme Project Management
Phone: (713) 385-1118
E-mail: dmek@extreme-pm.com



JOVIAL GOT YOU PUZZLED?

STSC JOVIAL Services Can Help You Put the Pieces Together With:

- SPARC Hosted-MIPS R4000 Targeted JOVIAL Compiler
- Windows 95/98/ME/NT (WinX) Compiler
- SPARC Hosted-PowerPC Targeted JOVIAL Compiler
- 1750A JOVIAL ITS Products
- Computer-Based Training
- Online Support
- Use of Licensed Software for Qualified Users

Our services are free to members of the Department of the Defense and all supporting contractors.

Just give us a call.

If you have any questions, or require more information, please contact the Software Technology Support Center.



JOVIAL Program Office Kasey Thompson, Program Manager • 801 775 5732 • DSN 775 5732
Dave Berg, Deputy Program Manager • 801 777 4396 • DSN 777 4396
Fax 801 777 8069 • DSN 777 8069 • Web Site www.jovial.hill.af.mil

