

What is iterative development?

Part 3: The management perspective

Ian Spence
Kurt Bittner

May 15, 2005

from The Rational Edge: Iterative and incremental development requires project managers to consider each iteration as a separate project, to be managed in the context of the final deliverable. This final installment in a three-part series explores how the proper perspective on project and quality management issues guides a development team to success.



What does it mean for a project to work in an iterative and

incremental manner? In this series of articles, we address this question by examining the various perspectives of those involved with the project. In the [first article](#), we defined what we mean by iterative and incremental development and examined what this way of working means to the core development team producing the software. In the [second article](#), we focused on what it means for the customers when a project chooses to work in an iterative and incremental manner. In this final installment, we consider the effect of iterative and incremental development on the projects' management teams.

Once you have the business and the development teams working collaboratively in a series of shared iterations, focused on delivering business value back to the business, you need to consider the management perspective and why it is important.

First, imagine a software development project without any management guidance. (Perhaps you've worked on one.) Such projects usually suffer from a lack of medium-to-long term direction; progress resembles a random path and few team members are sure where they are going. Without mapping out iterations, you have very little idea when the project will be finished, what resources will be needed to complete the project, or when these resources will be needed. Without

estimates to help judge the cost of the project, it is difficult to get the customer and sponsor to commit resources for all but the most trivial project, and there will be little to indicate how the project is progressing toward the goal of delivering a solution.

It is very tempting to fall into the idealistic trap of thinking that if a team is committed to a goal, they will automatically self-organize and fulfill all of their commitments to the organization as well as deliver high-quality software rapidly and effectively. In reality, even the best teams need some oversight to make sure that day-to-day work is progressing toward the longer-term goal. More importantly, it is the task of management to bring together the team that can do the work.

It is hardly accidental that poor management is the root cause of many project failures.¹ It is tempting to believe (as many non-managers do) that managing is little more than tending to the needs of some distant bureaucracy, and that the role of the manager is to keep the bureaucrats at bay while the rest of the team does the work. In fact, much mediocre management does, in some way, resemble this picture. But if this is all the manager does, the project will probably fail.

Management is more than note taking, schedule keeping and budget minding: Providing leadership and direction are essential to achieving results. Proper management affords clear answers to essential questions:

- Are we solving the right problem?
- Do we have the resources to deliver the solution?
- Are we working on the right things, right now, to get us to our end goal?
- Are we fooling ourselves into thinking that we can actually deliver a solution within the time and resources allotted?

Planning and measuring are not ends unto themselves but are tools that help the manager answer these questions.

The project managers' perspective

As we established in Part 1, an iteration consists of the application of the core disciplines of software development to produce a demonstrable, executable release of the product under development and ensuring that the product will be grown incrementally across a series of iterations, each building on the product and lessons learned from the previous iteration. This is illustrated in Figure 3-1, which builds on the development team, business analyst, and customer perspectives shown in Figure 3 of Part 1 and Figures 1 and 5 of Part 2.

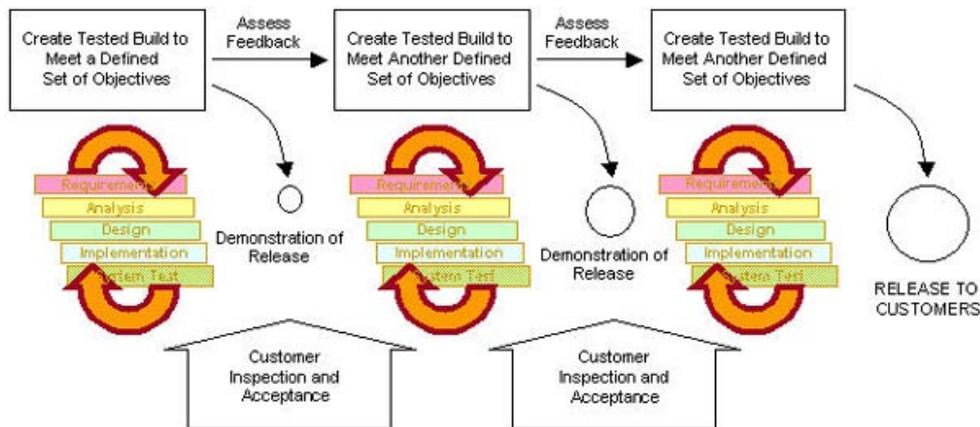


Figure 3-1: An iterative project from the project manager's perspective

From the project managers' perspective, each iteration appears to be a small, self-contained project during which all the disciplines of software development are applied to produce a release of the product that meets a specific, agreed set of objectives. This idea that an iteration is a small project agrees with most commonly used definitions of a project, such as:

Project: A temporary endeavor undertaken to create a unique product, service, or result.²

Iterations are certainly temporary, typically lasting between two to six weeks,³ and definitely produce a unique product in the iteration's demonstrable release.

At its simplest, the iterative planning process appears as a cycle of agreement, execution, and assessment:

1. Agree with the team the objectives for the iteration, including evaluation criteria, timescales, and constraints.
2. Agree on a plan for how the team will achieve the objectives.
3. Execute the plan.
4. Assess the achievements of the team against the initial set of objectives and evaluation criteria.
5. Assess the impact of the iteration's results on the project as a whole.
6. Start the next iteration.

From the manager's perspective, these activities occur continuously throughout the project, as shown in Figure 3-2, and form the basic pattern by which the work on an iteration is planned and managed.



Figure 3-2: Iteration from the management perspective

Managers also provide direction to the project as a whole, organizing work so that each iteration progressively contributes to delivering the solution. To do this they must plan, execute, and assess the project as a whole, and organize it as a series of contiguous iterations. The relationship between the two levels of management process is illustrated in Figure 3-3.

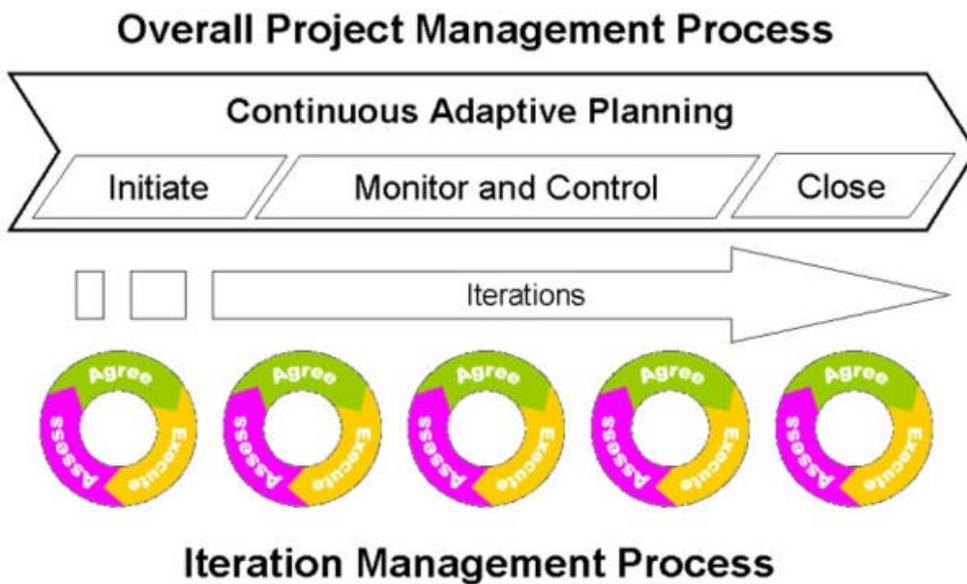


Figure 3-3: The overall and iteration project management processes

The planning, monitoring, control, and management of iterative and incremental projects is fundamentally different from that of non-iterative projects in terms of:

- The dynamics of the project team interactions
- The nature of milestones
- The handling of dependencies
- Measurement and metrics
- The resources required

- The degree of parallelism in the project work

One of the most significant differences lies in the way progress is measured. Instead of measuring progress by the completion of intermediate work products, such as requirements documents, analysis models, and design specifications, we measure progress in terms of how many scenarios have been developed and tested. As shown in Figure 3-4, each iteration will result in measurable progress toward delivery of the solution.

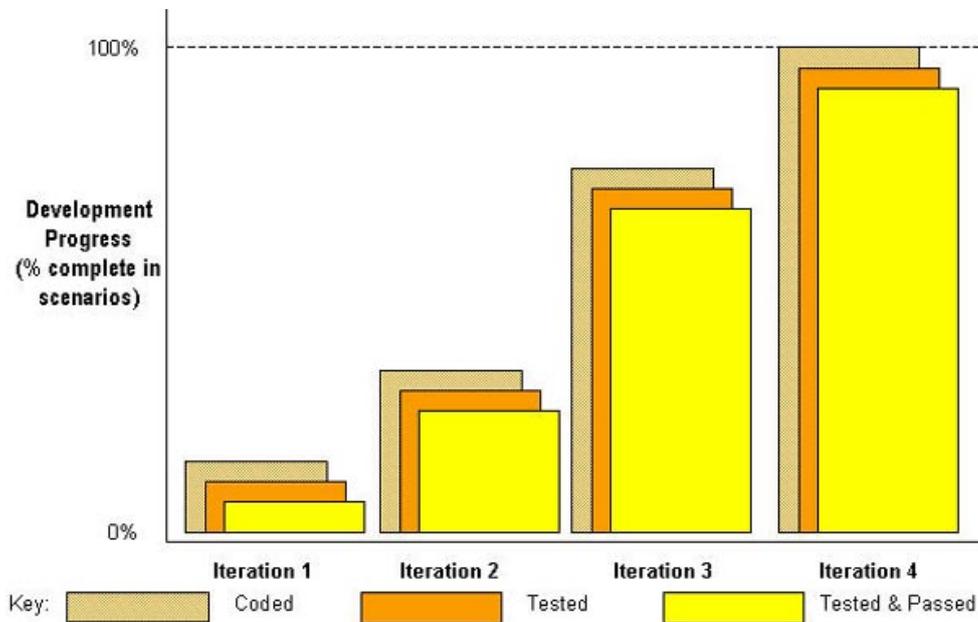


Figure 3-4: Measuring increments in successfully tested software

Shifting from subjective assessment of progress (often in terms of documentation produced) to objective measurement of progress (as measured by the amount of working software that has been produced) is the most fundamental difference between an iterative approach and more traditional approaches. Analyzing trends arising from objective assessment of progress at the end of each iteration enables the project manager to take control of the project and to make changes to improve the odds of project success.

Iterations enable the project to become a series of smaller self-contained projects, each of which is dependent upon the results and performance of the previous one. The feedback generated by assessment at the end of each iteration allows action to be taken by adjusting the plans for the next and all subsequent iterations.

The quality manager's perspective

These regular iteration assessments provide the mechanism for judging progress. At these minor milestones, progress is assessed against the goals and objectives set forth by the team in the iteration plan. One implication of this is that activity-based status reporting (What things are you working on?) is not as useful as achievement-based status reporting (What have you

accomplished? How close to being done are you?). With the iterative approach, it is hard to hide the truth for very long.

In truth, the typical approach of measuring activities rather than real progress only causes problems. The project team needs the flexibility to adapt their activities to meet the desired goal; the project manager cannot possibly foresee the future accurately enough to plan all the work needed to accomplish the goals. It is far more enlightened and liberating to give the project team goals to achieve and to empower them with the flexibility to respond to change. By constraining creativity and responsiveness, detailed activity-level planning actually endangers project success rather than ensuring it.

Objective measurement of project results during every iteration removes the need to assess project progress through subjective quality assessments of intermediate products. Instead of judging the quality of the requirements by insisting that they be *signed off* before anything can be built, you can evaluate requirements by taking the first ones available and driving them through the development cycle, within the iteration, to produce working, tested code. This provides objective evidence of their fitness for purpose, as they will have actually been used rather than just admired. The way quality is measured fundamentally changes. Quality measurement is focused on the regular iteration assessments that provide continuous insight into the project. Retrospectives on the process and working practices regularly capture the lessons learned by the project and engender a culture of continuous process improvement within the project. These observations can be immediately applied within the next iteration to improve quality and team performance.

Objective measurements are made in every iteration by testing the release produced, providing management and quality indicators, as well as measuring project progress. This continuous integration and testing results in increasing test coverage as the project progresses, as each iteration regression tests the requirements delivered by the previous iterations, as shown in Figure 3-5.

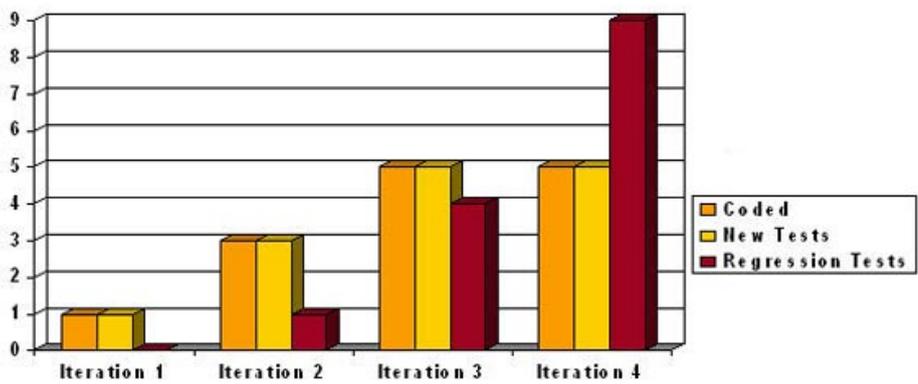


Figure 3-5: Test load increases iteration by iteration

The adoption of an iterative approach helps to improve the quality of the results delivered by:

- Reducing requirements misunderstanding early by forcing the requirements to be rapidly translated into things that can be observed and measured

- Reducing development risk early by proving technical approaches
- Accommodating and controlling change from the start of the project
- Providing a way to evaluate process effectiveness, schedules, team performance, and resource adequacy early in the project while changes can still be made

In an iterative approach, quality managers focus on the usefulness of products and processes, supporting the effectiveness of the team as a whole. Emphasis for quality changes from a focus on completeness to one of "fit for purpose." The quality manager must let go of the misconception that *"signing off on products and never changing them again provides higher levels of quality."*

This doesn't mean that things never finish -- only that the project team will not allow the declaration, or impression, of completeness to get in the way of their ability to actively work to reduce project risk.

There is a widespread misconception that iterative development never really completes the job; some people view the flexibility built into agile, iterative projects as an indication of work forever in progress. This is most certainly not true. There is an end goal to which all iterations drive: the final release. Each iteration is like a course of bricks in a wall: each iteration contributes to the end goal of building the wall, and the contribution of each iteration to the overall goal can be measured, just as the contribution of an individual course of bricks to the total wall can be measured.

Summary: Iterating from a team perspective

Adopting iterative and incremental development techniques is not a purely technical decision that only affects the developers and other technical people involved in the project. It represents a fundamental change in the way the project is conceived and progresses, a change that affects everyone involved in the project. Iterative development therefore requires changes in the way that the whole project team works and interacts. At the same time, it is not such a profound change that project management is no longer necessary. Instead, it simply requires a different and more adaptive management approach.

Iterative development is, at its heart, a team-based approach to problem solving and solution development. It requires all parties involved -- including the development team, the customer team, and the management team -- to adopt collaborative techniques.

From the perspective of the development team, the adoption of iterative and incremental development is empowering, enabling team members to actively and aggressively attack project risks and challenges in whatever they judge to be the most appropriate manner. Managing iterations by setting clear objectives and objectively measuring results (and not dictating activities) ensures that they are free to find the best way to deliver results.

From a customer and business team perspective, the introduction of clear, meaningful objectives, combined with the ability to review demonstrable results, allows those who will ultimately use the new software to take an active role in the project and share its ownership with the development team. Iteration has a profound and lasting impact upon all of the business people involved in the project and fundamentally changes the way that they specify, pay for, and realize the business benefits of software solutions.

From a management team perspective, each project is decomposed into a series of smaller projects, called iterations, each of which builds on the results of the previous one to incrementally achieve overall project goals. This segmentation of the project introduces regular, measurable milestones that keep the project on track while empowering the development team to create innovative and effective solutions, maximizing the project's probability of success.

Adieu

We hope you have enjoyed this brief examination of how iterative development supports and benefits the various participants in software development. As you will have gathered from this final article, we firmly believe that good management is an essential participant in the successful application of the iterative approach. This topic will be further explored in our forthcoming book *Managing Iterative Software Development*, to be published by Addison-Wesley later this year.

Notes

¹ See for example The Standish Group's annual Chaos Report available from <http://www.standishgroup.com>.

² According to the Process Management Body of Knowledge (PMBOK). See <http://www.pmi.org>

³ The actual length of an iteration will vary with the number of people on the project, which, in turn, varies with the complexity of the solution. Larger project teams require longer iterations to deal with the complexities of managing their communications.

© Copyright IBM Corporation 2005

(www.ibm.com/legal/copytrade.shtml)

Trademarks

(www.ibm.com/developerworks/ibm/trademarks/)