



THE SMARTER WAY

Getting Started with Iterative Project Management

March 2010

If embarking on your first iterative project fills you with uneasiness, and you have some doubts about whether you want to take the next step toward working iterative, this paper will help guide you on that journey.

Why Iterate?

Iterative development is undertaken to eliminate project risks early in the project, before they can have a chance to sink the project. By tackling risk explicitly, your project will be more likely to succeed. That alone should provide fairly powerful motivation. Other reasons for adopting an iterative approach include:

- To achieve higher quality
- To achieve faster results
- To achieve results more reliably, or sometimes just to achieve results at all
- To reduce staff frustration and turnover
- To achieve greater flexibility and business agility
- To reduce costs

The impetus for change can come from any number of sources: from senior management wanting to achieve better business results, from senior technical leaders wanting to resolve recurrent project problems, or from project teams that want to improve themselves.

There also needs to be a sense of urgency about the need to change; some crisis or significant opportunity is required to get the project team members to be interested in the extra work of learning how to do something new. Change is initiated for a variety of reasons. The reasons listed here are the major themes that lie behind most change initiatives. Often the reasons for the change are not well articulated or even well understood.

Unless there is a sense of urgency, the stimulus for change will never become great enough to overcome resistance. People become comfortable with the status quo, and it is difficult to get them to change unless they feel that they will personally benefit from the change. They need to believe that unless they change the way they work, bad things will happen, or in the case of opportunities, good things will fail to happen.

Potential Barriers to the Adoption of Iterative Practices

Understanding the potential barriers to change will help you make the right choices of timing, project, and approach. Some of the more important questions to ask include in the following:

- **How supportive is senior management of the change?** The measurements and milestones they establish can easily derail an iterative approach, as is the case when they ask even seemingly innocent questions such as “When will the design be completed?” or “When will requirements be signed-off?”
- **What is the scope of your authority to make changes?** How much of the development lifecycle are you responsible for? For example, the requirements might have already been specified in a format and to a level of detail that would make it more difficult to adopt an iterative approach.
- **What are the team’s feelings about the changes?** How enthusiastic is the team about iterating? To achieve the transition to iterative development, you will need the support of the team, especially the other members of the leadership team.
- **What else does the team have to do?** How many other projects and initiatives is the team involved in? If the team is not focused on and dedicated to the project, the transition to iterative practices will probably be slower and take more time and energy to complete.
- **What capability do you need to improve?** It is important to understand the capability of the team and how well the current capability supports the proposed iterative approach. For example, is there any testing capability in the team? Testing will be needed from the first iteration, which is often a problem in companies organized around the phases of a waterfall approach.

The ideal characteristics of an iterative project, and an actual project’s characteristics, are presented in Figure 1 on the next page.



Figure 1 shows that, ideally, your first iterative project should have:

- a co-located team,
- of less than 10 people,
- who are highly-skilled,
- and largely self-organizing
- who operate with low ceremony,
- and who are minimally dependent on outside teams,
- with highly-engaged business representatives.

These perfect conditions, in fact, almost never exist, but understanding them will help you choose projects that are better suited to an iterative approach.

When you start your first iterative project, factor the successful adoption of iterative techniques into the critical success factors for the project and the career development objectives for the team members. This is especially important if circumstances require an investment in training and mentoring the team to enable the transformation to take place.

1. As you become more experienced it is possible to start bending these “rules”, but when you are just starting out we recommend that you use these as selection criteria.

Conventional wisdom suggests that you should choose low-impact, non-critical projects to be the focus of early change efforts to reduce the change effort's risk and the potential for failure. The problem with this is that although these projects are lower risk, their non-criticality usually means that they are starved for resources and not considered "mainstream" enough to ever be taken seriously as success stories. This dooms the overall change initiative. Low-priority projects are not visible enough to engender the support needed to drive the change forward.

Instead you should:

- Look for a small number of must-do projects that have organizational focus and strong support
- Look for projects with a smaller set of stakeholders to keep communication simpler
- Look for projects with high internal visibility—ones that would make good success stories
- Choose projects that are short to medium-term in length
- Staff the projects with the best people available (respected leaders)
- Organize projects to generate short-term wins—divide work into iterations
- Keep the project size small in the first few iterations and then scale up
- Focus on choosing the right projects and committing resources to them

Choosing critical projects sounds risky, but the reality is that only critical projects will get the attention and resources necessary to succeed. The key is choosing projects critical enough to get resources but that can start small and then scale up in a controlled way, not becoming too large before the architecture and technical risks can be brought under control. Scaling up should be targeted for the Construction phase but not before.

Communicating the Goals of Change

To motivate the change in approach, you must be able to clearly and concisely communicate why a different approach is needed. If the business goal is to achieve better responsiveness to changing market needs, the goal for the iterative project effort might be something like be able to go from idea to released product in nine months (or less!). The more precise and measurable the goals, the easier they are to achieve.

It's important for you to do a couple things when you are communicating the reasons for the change:

- Be concise and articulate, and be able to explain it in a few minutes or less—The vision needs to be precise and specific, without sounding like empty slogans exhorting people to "do better." It should set specific goals that can be translated into criteria by which people can make decisions.
- Link problems to outcomes, such as "If we don't do X better, Y will happen"—Linking problems to outcomes is essential to getting a sense of urgency to solving the problem.

Being specific is also important. Generalization is easy but not very compelling. Ultimately you need to set specific goals. Being specific about the problem and its impact sends this message from the beginning. For example, “Our inability to deliver releases within X% of the projected date results in lost opportunities of Y million dollars per year” directly links the problem to its outcome.

You must communicate this vision at every possible opportunity, at every level of the organization. Everyone should understand what is being done and why. Learn to be able to concisely describe the vision in a few minutes and present it as an “elevator pitch”—think of a chance meeting in an elevator with a key decision maker. You have this person’s undivided attention for, at most, a few minutes. The vision needs to be crisp, concise, and compelling enough to explain in just these few minutes. Everyone today is busy; they don’t have time to carefully read pages and pages of reasoning and justification. Set it out for them, and your message will have a better chance of being internalized. The vision need not be very specific about how the change will be achieved, but it does need to be compelling.

Determining the Pace of Change

Every person and every project has a limited tolerance for change. People who have been successful with change initiatives in the past are more likely to embrace change, whereas people and projects that have a mixed or poor record of success with change will be more resistant to change. Trying to make too many changes too fast will be destabilizing and will make things worse, at least for a period of time. Most people have a threshold to the pace of change that they can sustain—try to push change faster, and the entire effort can stall and fall apart. The ability of the people to deal with change needs to be taken into account when planning change.

If the change is too large, people may lose hope that it will ever pay off and might abandon it. We have personally witnessed this many times, which is why we recommend an iterative approach to introducing change. This seems like simple common sense and no great innovation, but it remains a mystery to us why people so often try to push large changes in a single large initiative.

Expectations must be managed carefully. The tendency for teams to want to do everything immediately needs to be tempered; a sense of “proportion and pace” is crucial. The improvements must be driven fast enough to achieve the desired results as quickly as possible, but not so fast that the team gets confused or disheartened at the lack of progress caused by trying to do too much change at once.

Dealing with Skepticism

You will encounter skepticism and disbelief in the approach that you are proposing. You should be prepared to answer the skeptics because they are likely to become your biggest supporters if you can win them over. Everyone has seen grand new approaches that claim wonderful benefits but fail to produce results. The skeptics will ask, “What will be different this time?” The argument is based on the observation that every project manager starts every project with a new plan and the best of intentions, but the result is always the same: project slips, frustration, and failure. You must be able to answer the question, “Why should we believe you are doing something significant enough to affect the outcome?”

We have encountered this question often enough to have some thoughts on how to answer it. Appropriate responses include the following:

- “We recognize that change is inevitable, and we are taking an approach that recognizes that. We will set goals for our progress, we will measure ourselves against those goals, and we will adapt our approach in light of new information rather than assuming that we had all the answers at the start of the project, which you know is not true.”
- “We will focus on the big risks early, while we can still do something about them, and we will take explicit action to reduce those risks. If our first attempts do not work, we will keep at them until we have dealt with the risks.”
- “We will focus on delivering the most important things the business really needs in an agreed-upon time frame rather than delivering everything they want. This is an important change from how we have done things in the past.”

In short, you need to say and show how the iterative approach is pragmatic and deals with the realities of the world. Although real convincing comes only with the proof provided by real progress, most skeptics find the honesty of statements like these to be refreshing, provided that they are backed up with action.

Bootstrapping an Iterative Project

Starting your first iterative project is really just like starting any other iterative project:

- Start with a small team focused on the business and technical risks
- Challenge the team to start with a four-week iteration
- Adjust the iteration length in response to work and team culture
- Plan to deliver business benefit every three to six months
- Plan releases as well as iterations
- Start with a co-located team
- Start with a simple backlog-driven requirements approach
- Address the architectural risks before scaling up

Regardless of how large the project might appear to be:

- Start with the assumption that it is small—If you start big, it will stay big, and generally the larger a project, the more likely it is to fail.
- Do everything within your power to keep the project small—If you don’t fight to keep things small, they will naturally tend to get large.
- Layer the plans and keep them succinct and focused—To be comprehensible, plans must be small. Exploit the layering inherent in iterative projects to avoid unnecessary precision in planning the work.

The first iteration of your first iterative project will be the hardest to plan and manage; chances are you won't know exactly what to do and neither will your team, and this uncertainty is unsettling. It is also likely to be a new experience for the stakeholders as well as the project team, which will only compound the problems. To compensate for this, make sure that the iteration has modest objectives: whatever you do, don't set yourself and the team up to fail by setting ridiculously aggressive objectives.

It is common for initial iterations to run over their schedule. Allowing this to happen establishes a bad precedent. Keep an eye on progress during the iteration and be ready to scale back on ambitions even in the middle of the iteration to make sure that you will have time to assess results. This usually means moving some scope from the current iteration to the next in order to maintain the iteration time box. When you assess the iteration, you can decide whether to make future iterations longer.

Your first iteration will probably be over-planned. Don't worry; just keep learning and looking for ways to simplify the plans. The main trap to watch out for is overly It will take some time and effort to achieve the full benefits of an iterative approach. Your first evolution is unlikely to accomplish more than successfully producing the same amount of software in a non-iterative fashion, but there will be less rework, the project will be less risky, and the chances of delivering in the predicted timescales will be increased. As a result, it is important that you do not set unrealistic expectations for what can be achieved. Iterative development is not necessarily faster than traditional development, but you will be more certain of delivering the right result in an acceptable amount of time.

Maintaining Momentum

The more you iterate, the better you and your team will get. Each iteration results in management, planning, process adoption, and team interactions improving and iteration becoming easier. By struggling through the first few iterations, you and the team learn how to iterate and how to work together in an iterative fashion. Over time, the levels of planning, reporting, designing, and everything else will settle at a natural level suitable for the team and the project. You achieve this by reviewing results at the end of each iteration and adapting your tactics and plans, discarding or amending things that don't add value, and adding techniques as needed to resolve issues.

Understanding Where to Start

The foundation of iterative development is provided by a shift in project management focus away from creating detailed plans and measuring activities against these plans to results-oriented project management. Focusing attention on setting the right goals for each iteration and measuring achievements against those goals is the first and most important step in adopting iterative development.

Introducing effective practices in the areas of release engineering and change management (specifically basic versioning, baselining, and control over the things the project is producing) is essential to supporting this shift to becoming “results oriented.” The ability to manage change across iterations (to determine which changes should be addressed in each iteration) and the ability to create executable releases (which requires a reliable build process) are essential to being able to make the shift to delivering objective results.

You will need something to define and drive the work. The simplest technique is to use a Backlog, or a list consisting of features, defects and change requests. The Backlog is prioritized based on technical risk and business value, and then items from the Backlog are selected, developed and tested during an iteration.

If you are involved in new product development or are making larger and more significant evolutions to the software, you will need to add some additional skills, primarily the ability to understand needs and define requirements and requirements-driven testing. If you need to build a completely new solution, you will probably not have the luxury of building upon an existing architecture. You will need to have a more disciplined way of forming the architecture of the system and translating requirements into designs.

As the solutions become more complex, you will need to draw upon all these skills in a fully iterative approach that is able to dynamically respond to new risks and find creative solutions to new problems.

The following outlines provide a way to adopt an iterative approach in an incremental way:

First Improvement “cycle”

- Introduce Iterations as “time boxes” directed at demonstrable risk reduction
- Introduce continuous Integration (daily or more frequent builds)
- Introduce basic notion of a backlog
- Lead workshops to identify risks, and to plan iterations to address risks

Second Improvement “cycle”

- Introduce the concepts of scenarios and use cases
- Introduce key concepts about architecture and the relationship to risk reduction
- Lead workshops to identify architecturally significant scenarios
- Lead workshop on test case identification from scenarios
- Implement a formal change control policy

Subsequent Improvement “cycles”

- Formalize the definition of the architecture
- Integrate release engineering, release management, and test management
- Gather simple effectiveness metrics and begin managing based on them
- Achieve consensus on “style” issues:
 - Requirements
 - Test Cases
 - Design and Architecture
- Enforce style issues and baseline requirements, test cases and the architecture

The essence of this approach is to introduce techniques in each improvement cycle to address the specific needs of the team rather than adopting a set of techniques wholesale. This enables the project team to show results while they are improving rather than letting the improvement effort stand in the way of producing results. This overcomes much of the traditional resistance to change.

Learning by Doing

The advantage of this approach is that people learn by doing, and they learn only what they immediately apply. The conventional wisdom (as practiced in much of the industry) is that change is introduced by defining the new processes, selecting supporting tools, and then training everyone on the new processes and tools. As it turns out, this approach actually increases the risk that the change will fail—the opposite of the intended result.

Training is necessary but on its own is not sufficient and is often overemphasized at the expense of informal experiential learning. To understand why, let's consider five stages of learning:

- 1.1. **Knowledge**—Basic knowledge of the concepts and the facts
- 1.2. **Comprehension**—Demonstrated understanding of concepts
- 1.3. **Application**—Ability to apply concepts in simple contexts
- 1.4. **Evaluation**—Ability to apply judgment on when and how to apply concepts to more complex contexts
- 1.5. **Innovation**—Ability to extend concepts to new contexts

All that can be expected from a training class is to convey basic knowledge and maybe to provide some practice in applying the concepts. Time and experience are needed for the change to actually take hold in the day-today activities of the team. Only when this occurs is the change successful.

To achieve sustainable results, change must be driven by doing real work.

2. These are loosely adapted from Benjamin S. Bloom's Taxonomy of Educational Objectives (Boston: Allyn and Bacon, 1984)

There are several reasons for this:

- People don't like to change, but they will change if they can see that change leads to positive outcomes. "Positive outcomes" means different things to different people. For management, it might mean more reliable schedules and improved delivery capability. For the people on the project, it might mean improved career development or improved quality of life. Successful change initiatives usually require the majority of stakeholders to achieve some positive result. The sooner positive results are shown, the sooner support for the change builds.
- Even positive changes are initially disruptive; few situations are so bad that change is not initially destabilizing. In addition, most changes take some time to produce results. While the changes are underway, a reserve of "good will" is gradually consumed until results become visible. The longer the change takes to show results, the more "good will" is consumed. While "good will" exists, the organization can afford to be patient, but after it is exhausted, impatience takes over and the change effort typically falls apart. The problem is that organizations that need to change the most typically have the lowest reserves of good will. As a result, it is essential that results be demonstrated throughout the change effort to maintain the momentum of change.
- Real change only occurs when people's daily habits change; habits change only with time and practice. As a result, change based on "telling and teaching" almost never succeeds. Real change requires "doing."

Achieving improved results is not inconsistent with introducing change. In fact, achieving improved results while doing real work is essential to achieving sustainable results and is at the heart of the iterative approach. At every iteration assessment, lessons will be learned and acted upon to improve the project's performance and results. By its very nature, iterative and incremental development encourages learning by doing for everybody involved in the project, which enables the approach to support its own adoption and improvement.

The Role of Coaching

When a new way of working is introduced, it is important to support the transfer of knowledge and application of new ideas. Workshops are an effective way to do this because they enable people to learn in the context of doing real work, and they accelerate the rate at which people can be productive using the new techniques.

To feel confident in the change, people need to be able to focus on "doing" and not be distracted by "figuring out what to do." Experienced practitioners who lead people through the new behaviors, getting them to apply the new process by doing, must provide guidance and coaching. Having experienced coaches available to the project team provides confidence in the outcome by providing experienced resources for the project team—people who have already been successful on other projects. A sense of confidence in the outcome is essential to building resilience within the team so that team members can recover from the inevitable setbacks that occur in any change effort.

To facilitate experiential learning, coaches enable the team to focus on "execution" instead of process definition.

In addition, there are several other benefits of this approach:

- It builds support for the change by creating real success early.
- It lets team members focus on “learning by doing” rather than sending them to classes and hoping they can put the knowledge into action on their own.
- It builds expertise within the team so that over time the team can support the new way of working without external help.
- It reduces the risk of failures, delays, and quality problems resulting from the team’s learning by doing.
- It accelerates the learning process by eliminating much of the uncertainty, discussion, and trial-and-error associated with learning by doing.

Some coaching is usually necessary to enable project team success. This will vary from team to team, but typically follow-on workshops are used to develop skills needed to execute the iteration plan. For example, in the Inception phase, project teams will need to understand how to understand the problem and create a vision for the solution; a workshop led by an experienced facilitator is often the most effective way to make progress toward this goal.

Sometimes the best coaches are on your own team in the form of the more experienced team members. Encouraging team members to work together to build team skills that improve team results also reinforces team cohesiveness. This does not happen accidentally, however; you will need to encourage it and plan for it.

Conclusion

Adopting an iterative and incremental development approach is a fundamental change in working practices for the management team and everyone else involved in the project. Successful iterative and incremental development requires a progressive and adaptive approach to be taken to the management of the project and requires the whole team to embrace change and the continual improvement that this change will hopefully produce.

In any change effort, it is essential to demonstrate the value of the change as soon as possible to overcome resistance and build support for the change. The only way that can be done is by achieving the desired technical and business results quickly and efficiently. The fastest way to reach these results is to introduce the change as part of getting real work done; if the change is considered separate from the “real work,” it will never produce results. With the guidance and leadership of an effective coach, and with the support of management to measure and reward positive results and positive change, teams can improve their process while getting real work done. Process improvement and getting results should not be considered mutually exclusive.

To expand beyond individual projects, you will need enlightened but benevolent dictatorship coupled with the demonstration through real results to all involved that the future can be better. It also requires leadership, real leadership—not the phony slogans of motivational posters, but roll-up-your-sleeves, hands-on leadership from the front that shows that you have a stake in the outcome. No one is going to believe you if you sit on the sidelines cheering; you have to be in the game.

Iterative development is not hard, but changing the way that people work is. In this book we have provided you with the background information and the practical guidance necessary to deliver better results through your software development efforts. The next step is yours: you now get to put these concepts into action. We hope that the approaches and techniques we have presented in this book will help you and your organization to succeed and thrive by achieving the full promise of iterative development.

Europe

+44 (0)20 7025 8070
info-eur@ivarjacobson.com

Americas

978-649-2856
info-usa@ivarjacobson.com

Australia

1300 567 280
Info-aus@ivarjacobson.com

Asia

+8610 82486030
info-asia@ivarjacobson.com

Find us on the Web:

www.ivarjacobson.com

Ivar Jacobson International

Ivar Jacobson International is a global services company that helps software organizations transform and improve the way in which they develop software solutions as well as guide them in meeting the expectations of the business. Our consultants provide an environment of experiential learning to develop the right competency levels amongst all roles and functions by becoming an intricate coach and mentor to the entire team. We have a framework that we adapt to effectively define and communicate business and technical expectations across the organization as well as create collective responsibility by teams and individuals for project outcomes. We introduce a proven practice driven approach that is goal oriented, incremental and measurable and is highly successful with either an existing software project or the implementation of new systems. We support our customer engagements with a rich set of technology assets inclusive of training materials, practice guides, and tooling.