Abstract

Many Agile projects are doomed to fail before they even begin. Most projects fail not because of the technical ability of the team, but because project goals, objectives and context have not been taken into account when forming the project plan. Further, most teams new to Agile often lack the basic training required to be successful on their first Agile project.

Using a lightweight project initiation framework (an inception deck), combined with an Agile boot camp, teams can drastically improve their project’s chances of success. This will both ensure more Agile projects are delivered successfully, and aid Agile’s adoption into the greater software community.

Problem 1: Unclear Project Goals and Objectives

A real challenge with any project is ensuring everyone is on the same page. When sponsors, stakeholders, customers, and developers all have different ideas regarding goals and objectives, it is unavoidable that at the end of the day someone is going to be disappointed. Usually it’s the customer claiming “this isn’t what I wanted”.

Agile software development does much to minimize the chances of miscommunication. Iterative development and the simple act of continuously putting running and tested software in front of the client for feedback helps projects adjust course when necessary.

However, bigger project issues like scope, necessary interaction with external parties, and conflicting priorities from multiple stakeholders can still catch projects off guard if not dealt with and tackled early.

Believing there is consensus where none exists is what kills most projects.

Unfortunately this happens all too often. A project is scoped and budgeted six months to a year before the development is to begin. When development time comes, no one checks to see if:

- the goals of the project are the same,
- the technology has changed,
- the scope has changed,
- the priorities have changed, or
- the organization faces any new competitive threats.

The project starts under the illusion of consensus where it may not exist.

Solution: The Agile Inception Deck

The inception deck is a tool we use to help us kick off new projects, and shine a big bright spotlight on the most common, controversial, show stopping issues we see facing projects today.

The inception deck was created in the spirit of Agile software development – simple and light\(^1\). The goal of the deck is not to get bogged down in six months of pre-project planning and speculation. But rather obtain the same result in a much shorter period of time – typically two weeks.

The goals of the inception deck are to:

- Gain and sustain commitment for the project
- Resolve any potential conflicts on project goals or objectives
- Make sure everyone is on the same page
- Set expectations

\(^1\) The original inception deck was created by colleague Robin Gibson.
Upon completion, the inception deck is a snapshot of what everyone agrees the project should be at that moment. Pictures, graphs, and images all help visualize the nature of the project, and ensure everyone is on the same page.

The power of the deck is its clarity and speed of execution. In less than two weeks, we distill the essence of the project down to a handful of images and ideas that clearly communicate the relative size, complexity, and scope of the project.

At its highest level the inception deck can be summarized as the following exercises:

1. Why are we here?
2. The elevator pitch
3. Business vision
4. Objectives and scope
5. Organizational context
6. Logical scope
7. Technical vision
8. Risk management
9. Project estimates
10. Trade-off sliders

Each exercise (or slide) is designed to clearly communicate the driving forces, and constraints bound to the project.

The Why are we here? slide is a reminder to the assembled team what the business and departmental objectives are. It is so easy to get caught up in the day-to-day minutia of work that people often forget why they are there in the first place and who the company’s customers are.

The elevator pitch is taken from Gordon Moore’s book “Crossing the Chasm”. The point of the elevator pitch is to distill the essence of the project down to one compelling sentence. For [target customer] who [statement of the need or opportunity] the [product name] is a [product category] that [key benefit, compelling reason to buy] unlike [primary competitive alternative] our product [statement of primary differentiation].

Business vision is a fun exercise where the team is asked to picture the project as if it were advertised on a cereal box. Here the team is encouraged to think of their project as a product and choose those terms and phrases that would be most compelling to buyers.

Objectives and scope is perhaps one of the most powerful slides in the deck. When projects have multiple stakeholders, each is going to have a take on what the objectives and scope are. What makes handling these expectations doubly difficult is that while everyone will want the project to succeed, each may have a different definition of what success is. When teams start projects without a clear understanding between stakeholders on what the objectives and scope are (and more importantly are not) the project is in jeopardy before it even begins.

The Organizational context slide is about identifying which entities within the organization you will need to build relationships with and make a part of your project’s success. Sometimes your project community is bigger then you think. Not identifying these entities early can have serious repercussions later. The goal is to proactively engage these entities early and often and begin building bridges with them.

Logical scope is about getting our head around the size and complexity of the project. The goal is to create a high-level one page visual so that anyone can quickly get a sense of how big the system is.

The purpose of the Technical vision is to get a sense of the complexity, architecture, and technologies involved in the implementation of the system. This slide is important in raising any technical issues the project may face.

Risk management is one of those un-sexy, naysayer exercises that many people would be happy doing without. While I can sympathize, I have found talking about projects risks an excellent forum for bringing to light many of the things that can potentially sink projects if left to linger in the background unchecked. Many executives do not understand the inherent risks and issues that come with software projects. Talking about the project’s biggest risks, and how the team is going to mitigate them while we are all calm and rational, is much easier than doing so after.

Sponsors are often only interested in two numbers when evaluating a project: how much is it going to cost and when is it going to be done. The Project estimates slide is about presenting the team’s best guess to those two important questions. With this information they can set their bosses’ expectations and allocate capital accordingly.

The trade off sliders are about bringing clarity to what trade-offs are to be made during the execution of the project. Every project has competing (and sometimes conflicting) goals. The fact-of-the-matter is all our customers want their projects built yesterday, to the
highest level of quality, for free. As much as we would like to do this however we know we can’t – some tradeoffs are going to have to be made. The most common tradeoffs that teams need to make are around time, budget, quality, and scope. But, there also may be others such as team satisfaction, training, or fun. Whatever your project tradeoffs are, represent them as sliders and have a frank discussion on what the project priorities are. Ask if date is the most important thing. What about money? What about quality? Then get them to rank the priorities (I use sliders on a bar) from highest to lowest. Make it clear that no two tradeoffs can be at the same level.

High level themes

The point of the inception deck is to have difficult conversations early. Don’t expect clients or management to have their house in order. Projects start every day with conflicting goals, under resourced teams, and unclear expectations. Raise these issues early and confront them head on before the project begins.

Keep the inception deck light. The point is not to produce copious amounts of documentation. I use power point, often limiting each exercise to a single slide to remind myself and teams that brevity and clarity are key. ‘Less is more’ is an inception deck operating principle.

Finally, the inception deck is a living document. It is not created at the beginning of the project and then discarded. It is a living, breathing project artifact that needs to be updated and revisited periodically when major aspects of the project change.

Problem 2: Inconsistent Application of Agile Practices

Agile software delivery works best when the entire team is pulling in the same direction. When everyone applies the practices with similar effort and spirit there is greater team harmony.

When some team members apply the practices and others don’t, productivity and quality suffer. The team becomes disharmonious. Left unchecked this can lead to dysfunctional behavior and team conflict.

To understand and experience the full benefit of Agile, teams need to be working together as one - pulling in the same direction.

Solution: The Agile Boot Camp

The Agile boot camp is an intensive short term engagement where we put newly formed teams through a series of tutorials, labs, and role playing scenarios to demonstrate how Agile software delivery works. It is ideal training for teams new to Agile software delivery who need a facilitated and safe training program in which to explore, ask questions, and discover what could potentially be a very different way of developing software.

The boot camp is where we:

- talk about peoples’ roles and responsibilities on Agile projects
- demonstrate how the Agile delivery practices work
- set group rules for acceptable behavior.

The boot camp is our ‘come to Jesus’ event, where we wipe the slate clean, forget about the past, and look forward to the future. It is where we as a team, sit down and as one talk about how we are going to work, live, and play together while delivering high quality working software.

Is basic training really necessary?

I wish it wasn’t but unfortunately it is. While universities and other teaching institutions are slowly becoming more aware of Agile and its respective techniques and practices, the practices are not yet pervasive enough in industry to not require additional training.

Take unit testing for example. We have had the ability to write executable, automated tests proving that our code does what we think it should do for many years now. And yet I still meet developers who don’t unit test their code. It’s criminal.

Too many teams don’t incrementally improve the design of their code as their architecture evolves – and they wonder why eventually they find it difficult to add new functionality to the system.

While the days of writing sloppy untested code may continue for a while longer in our industry, I would rather leave that work to our competitors. Those days are long gone on Agile projects.
The four day boot camp

The most common boot camp I have used in the past for introducing teams to Agile delivery techniques is a four day boot camp. I like four days because it is just enough to cover the basics before plunging into real world iterations (where the real learning takes place).

I have used this boot camp for mostly co-sourced consulting engagements with clients. Our most popular mode of operation at ThoughtWorks with clients is a co-sourced model where teams are a mix of client and ThoughtWorks people. Clients like the 50/50 split as it gives sufficient coaching and training time to those new to Agile.

The agenda for a four day boot camp looks something like this:

The Whole Team (2 days)
- Introduction to Agile
- Roles and Responsibilities
- Release Planning
- Team practices

Developers (optional for others) (2 days)
- Unit testing
- Refactoring
- Test driven development
- Continuous integration

The first part of the boot camp covers material relevant to the entire team. As an intro we will usually say a few words about Agile, and then move into the more interesting stuff around what people can expect to be doing day-to-day on the project. Release planning is a big section that covers how Agile projects approach project planning. Team practices is a catch all for those practices we would like everyone on the team to be aware of (i.e. the daily standup).

The second part of the boot camp is more focused on the software engineering practices we need our developers doing if we are going to work Agile. Here I like to review what I call the non-negotiables. By that I mean if the team doesn’t follow these practices, we are really in trouble.

Without stating the obvious you will need to alter your boot camp to meet your project’s requirements. For instance I continuously need to adjust the roles and responsibilities portion of the boot camp for each project. Some projects have roles like technical writers, or user interface designers. Their roles, how they contribute, and their interaction with the team needs to be discussed and articulated.

The non-negotiables

When it comes to the Agile development practices I reserve the developer portion of the boot camp for labs and tutorials on these software development non-negotiables: unit testing, refactoring, test driven development, and continuous integration.

Why not include other practices like pair programming, and on-site customer? It is mostly a matter of style and prioritization. I find when introducing teams to Agile, there are some practices I am prepared to be flexible on. Take pair programming for instance. I am a big fan of pair programming and believe it is in the team’s best interests to pair program continuously throughout the day.

If I am getting strong push back, however, from the client (due to the appearance of two programmers sitting together at the keyboard) I will indicate that I have no problem with developers working independently, save that when they run into a difficult problem they are able to call on their colleagues for help – to which every manager invariably agrees.

If the team doesn’t unit test, refactor, start moving testing into the forefront of the development life cycle (TDD), or continuously integrate – all the other practices begin to fail because of their dependence on these four.

When executing your boot camp, be prepared in knowing which practices you are able to flex on, and which you are not. Each project will have different challenges and stress points, and you will need to adjust your message accordingly.

Don’t preach

One word of caution with the boot camp and introduction of Agile in general - if you have been given the honor of showing others what you perceive to be better ways of delivering software, do so with humility and respect for your audience - no one likes to be preached to. My best boot camps have been those presented in the context of ‘We have some ideas
we would like to share with you on how we think we can work together effectively. We would love to present those to you now to begin a discussion and seek your input into how we can most effectively set this project up for success.’ You need to be sincere and genuine when communicating this. Don’t just say it – believe it because that is really what Agile is all about.

This last point I raise because I have seen some consultants (yours truly included) turn audiences and teams off Agile because of how righteous we seem when discussing the practices. On the other hand, you can’t come across as too wishy washy else you will lose all credibility with the client as it looks like you don’t stand for anything.

You will know your situation best. All you can do is share what you have seen work in the past, and try to get buy in from the team. Trust your gut, speak from the heart – but don’t preach.

**Conclusion**

Asking the tough questions posed by the inception deck early can catch potentially fatal project risks before they blow up into show stopping problems. In hind sight, several failed Agile projects I have been a part of would have benefited from completing the inception deck. I recommend teams kicking off new projects consider using the exercises outlined in the inception deck to ensure their projects aren’t doomed before they begin.

After applying the Agile boot camp on several projects, I would recommend every project begin with one, regardless of the team’s previous experience with Agile software delivery. Getting team members together and discussing how they would like to work together is always a good idea. While the agenda may vary slightly from project to project the goal is the same. To form a tight bond with team members, and collectively discover the best way forward working together.