

How to Hit the Ground Running

A New Look at Iteration Zero and Software Development

How many times have you been involved in a project where management wants more architecture documents produced... but there just isn't enough information to produce the documents? How about when management wants to you to start coding the moment the construction phase begins?

While problems like this plague development shops, it doesn't have to be that way. This white paper presents the perfect plan (based on the Agile method of Iteration Zero) to make sure that when construction begins, coding really does begin.

Getting Out the Door

We are going on a journey. Whether our journey is a software development project or travel to a new place, preparing for a journey should follow a process. Journeys that adhere to a process right from the start are much more likely to be enjoyable, problem free and capable of delivering better results.

For purposes of this article, let's say that we want to travel from Toronto to New Jersey (don't ask why, just bear with me). However, there are several things that we do not know about this trip. In fact, we don't even know all the things we don't know about yet. And, if we think that we know, we are only fooling ourselves. So what do we do? We pull out our map, and begin to look at the most direct route from Toronto to New Jersey.

First we need to answer some basic questions such as "How will we get there?", "When will we leave?", "When will we arrive?" and "Why are we going?" Mentally, we start a high-level planning session to answer these questions. We eventually determine that we will be traveling on foot because the other options are just too expensive for our limited budget. We also determine that when we travel, we should arrive at our destination in about 30 days. We know that we are going to New Jersey to reunite with some old school buddies. When our visit is complete, we need to trek back to Toronto. This means that the total trip will span 67 days.

Then we start to drill down into the next level of detail. We need to cross Lake Ontario because if we walk around we will add an additional five days. We know that we are very good at swimming and coincidentally were born with webbed feet. We also know that we can swim across Lake Ontario in 3 hours. We plan our route across the lake and make plans to eat at Sally's Diner in Buffalo, New York. We plan to rest in Buffalo and spend the night.

We then continue this level of thinking and planning as we travel through several states, rivers and finally over the Appalachian Mountains to our destination. After considering the potential for foul weather and our actual ability to make this trip, we realize that the trip will not take 67 days, but actually 84 days. This means that we need to start walking, or swimming, no later than the last week of March to arrive the first week of May. But, it's already the second week of April. So, now we revisit the plan to include running instead of walking. Such is life.

Now that we have all of these plans, the next step is to actually physically setup for the trip. The objective of this preparation phase – which takes an entire week – is to ready ourselves for the long journey and allow us to walk out the front door with confidence. It includes shopping for eight pairs of good running shoes, two pairs of hiking boots, a wet suit, some cloths, toiletries, a new back-pack and many other things. Next, we start a more detailed mapping of our exact route onto a map. Then we start conditioning our bodies for our impending, physically taxing experience. Without performing each of these activities, we increase the probability that we will be late for our reunion... or worse, not even make it at all.

We now have everything we need. We are ready to walk out the front door.

Iteration Zero is the work that needs to occur before actual software construction begins. This is the phase that occurs after project denition, but before construction. Done correctly, Iteration Zero prevents getting stuck in analysis paralysis and the eternal waiting-to-begin phase.

While some teams attempt to deliver useful functionality at the end of the first iteration, it is often more pragmatic to first focus on building a foundation before delivering any functionality. This is particularly true for projects lacking clarity on requirements or larger, more complex applications.

Getting Down to Work

So, what does all of this have to do with Iteration Zero and software development? The whole process described above is all about managing risk and approaching the goal properly. We want to plan enough to have a successful journey without stifling ourselves with too much overly complicated process. Process is supposed to enable, not disable.

In the first stage of planning for our journey, we identified where we want to go and how we think we will get there. In software development, this can be thought of as business definition and understanding the business objectives. In the second stage of preparation for our fictional journey -- studying a detailed map and the purchase of travel gear -- can be thought of as Iteration Zero. This is where the team's virtual assembly line is prepared. Finally, the third stage of swimming and running is the construction phase.

Iteration Zero sets up the groundwork for construction by implementing the reusable patterns and components in the application architecture and preparing the environment for a development team to be effective. This phase helps to remove the traditional Just-In-Time environment setup that occurs in the construction phase. By adding an Iteration Zero, you are essentially allowing for time to be utilized more effectively and predictably during the construction phase.

One perception about the phrase "Iteration Zero" is that it implies an agile style of development. This does not necessarily need to be true. The practices in "Iteration Zero" have been used successfully with several different development methodologies and it works equally well.

One of the major dependencies for an effective Iteration Zero is awareness of the success criteria of the project or program. This can be as simple as a bulleted list that is taped to the wall for everyone to see. These success criteria need to be gathered from and agreed upon by the business. They also need to be understood by the technical team(s) in order to start the process of alignment.

Other benefits of Iteration Zero include:

- A head start on working out the kinks with team dynamics and processes by actually practicing the team interaction in a sandbox.
- The opportunity for different practices within the team to gain momentum.
- Interdependent work can be planned and completed before the construction team hits the ground running.

The content of Iteration Zero can vary significantly from project to project and business domains. For example, a checklist for an Iteration Zero phase might include:

- Create an install package for each development machine to use
- Setup the development VM servers (IIS, DB)
- Setup the QA VM servers (IIS, DB)
- Setup and Configure the VM Build Machine
- Create a solution in Visual Studio and add it to source control
- Create projects and directories within the solution for organization
- Integrate the application framework
- Create and/or integrate Security, Logging, Session Management
- Design and generate the domain model
- Identify the central repository for business documents
- Identify and implement issue tracking
- Identify and implement task tracking
- Identify and implement change management tracking
- Create Style Sheets
- Create Master Page(s)
- Walk thru and QA the assembly line
- 3 day developer workshop (standards, approach, training, process, ...)

Think of Iteration Zero as the heavy lifting phase for aligning IT with business objectives.

This list can be more or less detailed based on the current practices of the team and/or organization. It needs to be tailored appropriately and prioritized by dependencies so that the work can be accomplished in the allotted timeframe.

Iteration Zero and Technical Debt

There is always the potential for unexpected curve balls to be thrown at the team. Typically, the team will react by cutting corners to maintain time-to market (TTM). They may say, “We’ll come back to that later.” Well, how often does that actually happen? Frequently, right? Right out of the gate, with a seemingly fresh Version 1 of your project, the organization has acquired TD and possibly some design debt.

One major goal of Iteration Zero is to provide alignment between the business and IT on the acceptable amount of TD that can be acquired. During this Iteration, business goals that influence the direction of Iteration Zero will come to the surface. This may include incurring some TD to accommodate goals such as TTM or preserving of investment capital. The team may also determine what short-term and long-term debt will be tolerable based on the cost.

Although TD is a topic that can be discussed at length in its own article, there are several key elements in Iteration Zero that can accommodate the tracking, scoring and measurement of TD.

For example, making sure that the SDLC can accommodate paying off the TD or using the correct defect tracking system to gather the TD elements as they occur. Keep in mind, however, that while the business may see some TD as tolerable and necessary, the technical teams usually strive for a zero-debt footprint. So, if TD is not tracked, it will be forgotten or lost until you start feeling some of its pain.

Avoiding the Gotchas

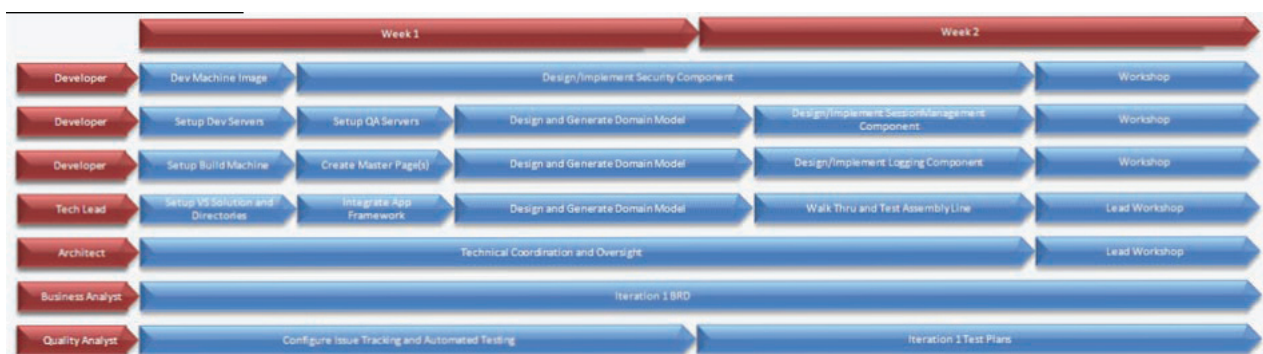
The most common gotcha of Iteration Zero is that it is never introduced into the SDLC. It is essentially forgotten. Barring that, there are other gotchas that typically arise while trying to fine-tune this phase. For example, the deciding on the level of detail requires particular skill and finesse. You do not necessarily need a project plan, but a release plan for these tasks is definitely a good idea. This allows the dependent tasks to be identified and managed.

Another gotcha is not putting the appropriate amount of detail into the mix which can lead to a smearing and overlapping of JIT tasks with planned tasks. This is a common culprit of the actual time spent being greater than the estimate.

A case in point: Sally is developing a tax component that will interface with some legacy web service that is now being hosted on a development server. However, when she tries to call that web service, it doesn’t exist yet because the development server didn’t integrate the suite of web services yet. Yet another ugly unplanned JIT development task is upon us. How many times have you seen this pattern? I have seen and heard about it approximately 8,376 times.

So, now the question is, “how will I know when Iteration Zero is done?” Iteration Zero will be “done” when the architect and Technical Lead can both say that the team can be fully productive on Day 1 of the first construction iteration. They should be able to march a team into an environment and sit them down with all of the tools and knowledge they need to get started.

What would this look like? Below is a sample Release Plan.



One of the downsides of not including an Iteration Zero is increased project risk to Technical Debt (TD) that can occur even before the first release.

Creating a Forum to Ask Questions and Learn

One of the most important elements for executing Iteration Zero is the Workshop. The Workshop allows the team to see how the software is going to be developed from a process perspective. Additionally, it helps the team become familiar with the technologies, components and patterns implemented and added to the code solution/project. This begins the process of the team becoming immersed into the construction process. Similar to the TD mentioned earlier, I also like to use the phrase "Process Debt." This is a negative side to process that causes pain and inefficiency that can be partially tested and corrected in Iteration Zero.

As illustrated by the Release Plan, in order for the development and quality teams to be effective in Iteration 1, the Business Analyst(s) should begin creating the Business Rules Documentation (BRD). Iteration Zero provides them with a jump start from both the development team and the quality team; both of which have a dependency.

If needed, the QA team can also have test scripts available for developers to unit test against. While I have seen this work effectively on some projects, I have also witnessed this not really making sense on others. Your call.

Iteration Zero can also be used to identify risky technical aspects of the system. These risky elements should be studied to form a mitigation plan as well as executed in Iteration Zero to learn as much as possible, as early as possible.

Conclusion

Although there are ways to prevent a project catastrophe, most of us have still seen at least one project land in the dumper for a myriad of possible reasons. While there are elements outside of delivery that can hurt a project, we can focus on this one piece and remove as much negative probability as possible from delivery.

Adding and enforcing Iteration Zero is a great next step to achieving that. Do everything that you can to avoid a death-march up-stream. Your next journey can actually become faster, more predictable and provide much more value to the business and the delivery team by thinking ahead when you plan your next Iteration Zero.

About the Author

David Katauskas is an accomplished software veteran with deep expertise in application architecture, framework development, system design and development process enhancement.

In his role as Enterprise Software Architect for a custom software development firm, Geneca, David plays an instrumental role in the adoption and success of Geneca's unique Getting Predictable best practices. David leads educational workshops on architecture and business/IT alignment and is a member of Geneca's Architecture Review Panel. To learn more about Geneca, visit www.geneca.com.

About Geneca

Chicago-based custom software development firm, Geneca, helps its clients meet their business challenges by bringing predictability to the software development process. Getting PredictableSM, Geneca's pioneering approach to Requirements Definition and Management, has an outstanding success rate in helping its clients drive clear business alignment by identifying project objectives and success criteria. Learn more about Getting PredictableSM and Geneca's other software services at www.geneca.com.

Additional copies of this whitepaper are available at www.geneca.com.



Geneca Headquarters
1815 S. Meyers Road,
Suite 950
Oakbrook Terrace, IL 60181

Voice: 630 599-0900
Fax: 630 599-0908
Toll Free: 877 436-3224
www.geneca.com

General: info@geneca.com
Sales: sales@geneca.com
Careers: jobs@geneca.com
Press: press@geneca.com