TARKENTON

IDEAS AT WORK

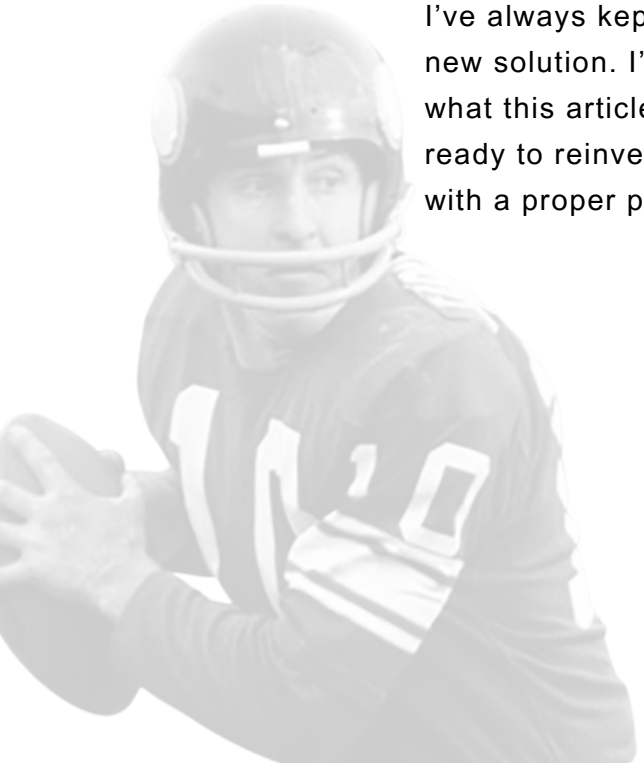
The Siren Song of Agile Development

Takeaways

- While Agile Development is popular, it's not right for every project
- Be careful that your project doesn't end up with a lot of "technical debt"
- Consider a blended approach that sets a firm foundation first

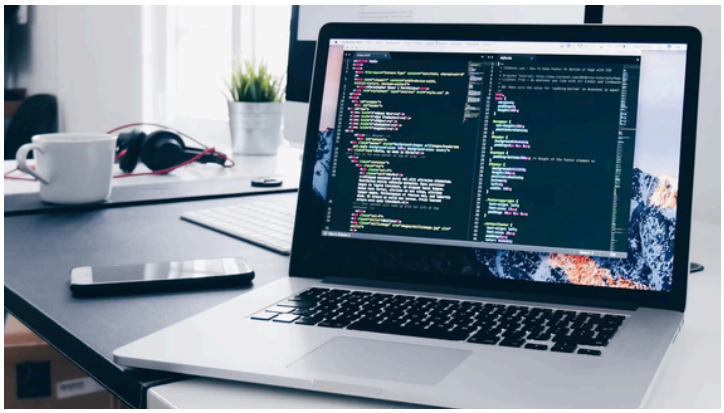
As a quarterback, I was a scrambler. When the play broke down, and the defense was coming at me, I didn't give up. So I would take off running – but not necessarily to try to run forward for yards, but instead I was trying first and foremost to buy time for a successful pass.

That scrambling was such an important part of my game, but people often forget that we never called a play that said, "Fran scrambles." We always had a plan for what we wanted to do after the snap, and the scramble was never Plan A.



I've always kept that mindset. I'm ready to adjust my plans and look for a new solution. I'm ready to scramble. But I always have a plan first. That's what this article is about. In any business, any project, it's important to be ready to reinvent yourself, to change and adapt. But you also need to start with a proper plan. It's not one or the other. It's both.

The Siren Song of Agile Development



If you are involved in software development (and what business doesn't need software development?), the magic word seems to be: Agile.

The idea of being Agile certainly has merit. Being nimble can help businesses adapt to an ever-evolving market environment or changing economic headwinds.

In theory, the popular software development framework called “Agile” offers software development teams the benefits of speed and flexibility.

But in practice, Agile may not be the magic bullet you’re looking for. In some cases, Agile can be beneficial, but it’s not the best fit for every project, and there are traps an agile project can fall into if you’re not careful.

If you’re thinking about adopting Agile, or you’re working with a development team that uses it, here’s what you need to understand before you proceed.

What Is Agile?

Agile is basically a project development philosophy that advocates breaking software development projects into small and manageable increments or iterations known as “sprints.” Each sprint involves planning, developing, testing, and reviewing a small piece of the final product. Proponents of agile tout flexibility, collaboration, and small, rapid software releases as key benefits. One of the key concepts of Agile is that customers are able to provide constant feedback while development teams adapt quickly to these changes.

However, one of the biggest challenges with Agile is managing the balance between flexibility and control. Too often, businesses or teams using Agile will start development without understanding the entire scope of the problem they’re trying to solve. Using a construction analogy, an Agile approach might be: “Let’s get the first floor built and we’ll figure out the upper floors later.” Trouble is, once you've built the first floor, it's costly and cumbersome to change it later, once you realize you need a different foundation or architecture or design to support the floors above. The same can be said for software development. That idea of adjusting as you go along is where you may experience issues with Agile.

Do It Fast or Do It Right?

One of the most common conceptions of Agile is that “it’s good because it’s fast.” The entire methodology is based on the idea of doing each small specific task as quickly as possible to keep moving to the next task.

But faster doesn’t always mean better. In fact, when companies experience problems with an Agile approach, the biggest issue that we see in practice is that speed can compromise the quality of the work as well as the documentation.

Scope creep is another common issue with Agile. One of the initially appealing aspects of the Agile methodology is that clients or stakeholders can decide to make changes to the product or project anytime throughout the process. Frequently, however, these change orders are out of scope. They can be done, but the new work extends the timeline. A project that was set to take six months is now stretched out to nine months or maybe a year. Not to mention the costs involved with the additional work. Stakeholders don’t always fully understand that extended timeline or the extensive extra costs until they’re billed for the extra work.

The role of the “scrum master” can also introduce challenges. The scrum master is the communication leader. He or she uses tools like the daily stand-up meeting to keep team members on task and on schedule. Their primary job is to make sure all the tasks are completed and that sprints are moving on schedule.

But they are not there to question why a task is taking a specified amount of time, or how each individual task fits into the entire project. The scrum master typically is not a developer, and therefore doesn’t have the technical expertise to judge how long a task should take. The scrum master is only focused on each individual short-term task.

When everyone’s goal is to get one task done and move onto the next one, it risks becoming an assembly line approach – create a piece of code and move on, then continue to do that until the software is done. But building software is not an assembly of parts; it is a creative process, and any new piece of software is part of an overall ecosystem. With so many other applications and programs connecting in some way, the assembly line approach just doesn’t work.

If a project falls into this trap, the result is that your end product often comes with a lot of "technical debt" – meaning more IT work that will eventually have to be done over time, either to revise existing architecture to accommodate other needs or to scrap work that was done but ultimately doesn’t meet the overall project goals.

Incorporating Alternatives to Agile

Achieving a successful software project or product starts with the right leadership – people who understand the project, the problem, and the definitive goal – and clarity throughout the team on what those goals and solutions are. With leadership and vision in place, the right development strategy is often found in taking more than just one approach.

For example, for many scenarios, the best approach is often a hybrid model that combines the flexibility of Agile with more traditional project management methods. This might mean starting with a “Waterfall” project management approach to establish a solid foundation with clear requirements and architecture, and then applying Agile methods for the development and testing phases. This approach ensures you establish a clear end-point goal, yet still allows for iterative progress and stakeholder feedback.

If you’re using an outside development company, make sure you have an advocate who understands your company’s existing systems and who will keep your business interests in mind. It’s critical to have a development team you can trust, whether they’re inside your organization or an external partner. How do you know? Look for a team that's interested in your long-term goals, that asks about your product life-cycle, that is dedicated to understanding the full scope of your project, as well as your plans for the future, before they start the build. The right partner is not just building a piece of software, they are committed to promoting it, maintaining it and helping your system evolve and adapt over time.

Whether or not you use Agile, every project needs a project flow or a product flow. Be sure you know your end goal and establish a timeline and a map of how to get there. Even if your development team uses the Agile approach, take a little extra time to scope out your thinking before you start, and you'll have a much better chance of meeting your goals on time and on budget.

By: Myles McNamara
Principal Software Engineer

Myles serves as the primary technical architect and full-stack developer for Tarkenton. As a key part of our development team, he understands the underlying technology available and helps to determine the required infrastructure and code bases for each project, ensuring we build complex, secure, reliable, and scalable software solutions. Myles has been working with Tarkenton since 2014, and a full-time member of the team since 2020 as our in-house code, security, and infrastructure expert with full development capabilities.