



Agile Is No Silver Bullet Building Great Software Takes Much More

Agile is a very useful approach to software development but it can't be applied in a vacuum. Without solid experience, proper engineering discipline, proven business acumen, open communication, and smart, creative people who exercise sound judgment, Agile is little more than a buzzword. This white paper cuts through the hype and describes how successful teams actually approach the software development process.

Introduction

Can you spot a 21st century snake oil dealer?

It can be hard sometimes. The software industry certainly has its share of hucksters who will promise you the moon, charge you for the moon, and then deliver green cheese instead.

If only they chose more revealing company names like *Professor Miracle's Cure-all Rejuvenating Software Services* or *Van Helsing's Silver Bullet Solutions*.

Unfortunately, they don't, so it's up to you to spot them.

Although there isn't any sure-fire method for distinguishing passionate experts from hype-masters (other than hiring them and finding out the hard way), you should always be on your guard when software substance seems to take a backseat to software-speak. For instance, if you ask a development firm, "How do I know you'll build great software?" and they reply, "Because we're Agile", then something is wrong. That's not a real answer. They might as well say, "Because we have new computers".

Like a computer, Agile development philosophy is a tool—a means to an end—and like any tool, it's only as effective as the people using it. But when Agile gets hyped as a selling point, this fact often gets lost in the clutter and nobody stops to think about it.

Agile is a development philosophy. It's not an indicator of creativity, skill, or character.

Granted, Agile is the best development philosophy we know of, which is why we use it. And we spend a lot of time explaining its merits to our clients so they'll understand where we're coming from. But at the end of the day, process is no substitute for people.

It's people who develop great software for other people. They do it with hard work, good judgment, discipline, business acumen, sound engineering practices, and creativity. There really is no silver bullet for software development.

This white paper looks beyond the theory of software development—beyond process and beyond buzz words—to identify three essential elements that high-calibre development firms rely on to build great software.



Of all the monsters that fill the nightmares of our folklore, none terrify more than werewolves, because they transform unexpectedly from the familiar into horrors.

The familiar software project, at least as seen by the nontechnical manager, has something of this character...So we hear desperate cries for a silver bullet...

- Frederick P. Brooks, Jr.
"No Silver Bullet"¹



Getting Past the Hype

Developing software is challenging, complex work and to many people outside the software industry it can seem positively arcane. Unscrupulous firms often try to capitalize on this lack of knowledge by selling “silver bullet” solutions that look good on paper but ultimately fail to meet expectations.

Ask yourself, how trustworthy is a software development firm that submits a fixed bid in response to an RFP with vague requirements? What do you think will happen when the vendor realizes that their proposed \$50K system will actually cost \$150K to build? They’re unlikely to swallow the extra cost and deliver a first-class system (especially since they’ve already demonstrated a lack of competence with requirements analysis).

How credible is a firm that promises to bring in “perfect resources” on a day’s notice, when they’ve been advertising the same positions for months on popular employment websites? And why do some firms think that it’s okay to staff a project with people who have never worked together before? Software development is a collaborative, creative activity that yields the best results when team members are used to working with each other.

When it comes to Agile, why do some firms think they can train their developers in a few hours? After all, Agile is a unique approach to software development, with a challenging regime that differs from traditional processes. Be wary of any firm where people regard Agile adoption as if it were like changing a light bulb. It isn’t.

When is an expert an expert?

Watch out for firms with “Agile experts” who really only have a few months of actual Agile experience. Novice firms, especially, tend to make exaggerated claims.

Whenever you’re confronted with self-described experts, ask for credentials and see what happens. Ask a lot of pointed questions and of course, be sure to check references thoroughly. Websites like www.glassdoor.com, where employees offer frank, anonymous reviews of their companies, can sometimes give you valuable insight into a firm’s actual capabilities.

What It Takes to Build Great Software

If you want to find a high-calibre development firm that will collaborate with you to build great software, you should look for one that has three essential elements:

1. A designer’s approach to software development.
2. Proper software engineering practices.
3. The right people on the team.

Let’s examine each of these elements in some detail.

Agile methods are challenging to grasp because they are based on soft-side principles in sociology, psychology, and behavior. Because these nontechnical factors tend to make learning, practicing, and mastering agile methods somewhat difficult, agile methods can take some time to fully penetrate even a small organization and become institutionalized and internalized.

*- Rico, etc.
The Business Value of Agile Software Methods²*



A Designer's Approach

Good designers are, above all else, *practical*; they see no value in process for process' sake. To a development team building software, this means:

- **Being flexible**

The team tailors the development process to suit each project and client. Team members must take into account existing IT infrastructure, internal standards, approval processes, quality assurance processes, risk tolerance, organizational structure, and the relationship between the business and its IT department.

- **Learning along the way**

The team takes "development" at face value and is prepared to apply lessons that they learn in one stage of the process to what comes next.

- **Delaying decisions**

The team doesn't try to foresee every possible scenario and contingency—it's impossible to do. Instead, they know that change is inevitable and they make important choices at the last responsible moment when the most information is available.

- **Being pragmatic**

The team uses good judgment instead of slavishly following a rigid process. If something doesn't work, they change it.

- **Communicating with users and all other stakeholders**

Instead of relying solely on written requirements, the team works hard to determine users' actual needs and then communicate those needs to the other stakeholders. Since team members are in constant contact with stakeholders, they can quickly address problems and resolve conflicts.

At Architech, we've found that using an Agile methodology is the best way to implement our designer's approach to software development. If we eventually discover a better way, then we'll use that instead. We care about results, not buzz words.

Proper Software Engineering

Without proper software engineering, a project can go off the rails in a hurry. Planning, discipline, standards, and best practices are all crucial because even the smallest software application can be a very complex product.

Here are five critical aspects of software engineering that can contribute enormously to the success of a software project.

1. Iterative development methodology

Software projects tend to have changing requirements, so it makes sense to deliver functionality in stages. A development team starts by building a working piece of software with several high-priority features. Then the client

Because software development work is so mentally taxing, it becomes critical that people working on the project take active steps to make the project as simple as possible rather than needlessly complicated.

- Steve McConnell
*Software Project Survival Guide*³

Everything should be made as simple as possible, but not one bit simpler.

- Albert Einstein



has an opportunity to try the software and reorder the list of requirements before the team gets to work on the next set of high-priority features.

2. Architecture

Software architecture is about planning the structure of an application and determining how its components will interact with each other and with external systems. One key goal of software architecture is to focus development resources on building the right software for your organization, as efficiently as possible. In keeping with the designer's approach to development, architectural decisions should be made as late as possible (without affecting development), when the most information is available.

3. Test-driven development (TDD)

TDD is a proactive approach to quality assurance. Instead of testing for defects near the end of development, when it's very expensive to fix them, programmers can use TDD to eliminate most defects at the source. Here's how it works: A programmer writes a test based on the software requirements and then writes just enough code to pass the test. In this way, programmers simultaneously develop clean code and build an automated test suite that will alert them if any code gets "broken" later in development.

4. Quality assurance metrics

To properly evaluate the quality of a software program, testers must be able to measure and report on characteristics like security, scalability, testability, performance, extensibility, flexibility, robustness, and maintainability. Without quality metrics, the development team has no meaningful way to gauge the effect of changes to the software, which is clearly unacceptable from a business point of view.

5. Project management

As we've already mentioned, software development is complicated, so "flying by the seat of your pants" is not an option for a team that expects to build great software. Professional project managers (we call them Scrum Masters) must support the development effort with planning, scoping, scheduling, resource allocation, and whatever else it takes to keep the project on track.

The Right People

People who know *how* to develop software can follow instructions, but people who are capable of figuring out *what* to develop really add value to a software project.

Software development is a complicated, communication-intensive process because it occurs at the interface between business and technology. A successful development team must therefore include people who not only have technical expertise and business acumen, but also a genuine willingness to collaborate. (Lone-wolf hackers and unintelligible geeks need not apply!)

Having the best, most beautiful code in the world matters very little unless it does what the customer wants. It's also true that having code that meets customer needs perfectly has little value unless the customer can actually use it.

- Shore and Warden
*The Art of Agile Development*⁴



This sort of team is most likely to uncover your *actual* business requirements—and the requirements of end-users—and then translate those requirements into great software.

Individual team members should have:

- Good communication skills.
- Experience working with each other.
- Experience with the technology that will be used.
- Experience building similar real-world applications, not just maintenance or support experience.
- Agile development expertise.
- A commitment mentality—they should be results-driven.

It's no secret that creative knowledge workers are the key to building great software, yet many development firms still treat people as a commodity. These firms, often called *software factories*, are sweatshops where young staff members are expected to work long hours for relatively low wages until they burn out and leave. Such firms are implicitly committed to delivering software that's "good enough".

Conversely, a software development firm that understands how people can make a difference is a valuable *technology partner* and not a mere resource shop.

Choose a Development Firm with Values

At Architech, our focus is on *servicing people with technology*, not on serving technology *to* people. When we talk about our people-centric approach to software development, we often refer to our **five core values**:

1. Embrace change

In software development, change is inevitable. In fact, changes are often positive indicators that we're learning our way to the best solution. We'd probably feel differently if our development process was based on a rigid traditional approach but we think it makes more sense to harness change rather than fight against it.

2. Think big

Creative knowledge workers such as programmers, project managers, and software architects can achieve amazing results when they're given challenging problems to solve. Even better, they love doing it and clients love the results! We swing for the fences because we know that's the only way to get a home run.

3. Be open and collaborative

We're technology facilitators, so our job is to make it as easy as possible for you to improve your bottom line with great software. To do this job effectively, we have to learn from you and work closely with you throughout

If you find yourself concentrating on the technology rather than the sociology, you're like the vaudeville character who loses his keys on a dark street and looks for them on the adjacent street because, as he explains, "The light is better there."

- DeMarco and Lister
*Peopleware*⁵

Once people have an adequate income, motivation comes from things such as achievement, growth, control over one's work, recognition, advancement, and a friendly working environment.

- Mary & Tom Poppendieck
*Implementing Lean Software Development*⁶



development. A transparent, collaborative process ensures that we deliver software that actually meets your needs.

4. Do the right thing

As your technology partner, we're going to tell it like it is—even if it isn't necessarily what you want to hear. That said, if we disagree with you about some aspect of a software project, we'll disagree constructively by pushing back with advice and viable alternatives. We want to work with you on all your future projects but we know that won't happen unless we earn your trust by doing the right thing now.

5. Never fail a client

We set realistic expectations for each project and then commit ourselves to meeting or, preferably, exceeding those expectations. We will collaborate with you to develop great software that works the way you do.

References

1. Brooks, Frederick P. "No Silver Bullet: Essence and Accidents of Software Engineering", *Computer*, Vol. 20, No. 4, April, 1987.
2. Rico, David F., Hasan H. Sayani, and Saya Sone. *The Business Value of Agile Software Methods: Maximizing ROI with Just-in-time Processes and Documentation*, (Fort Lauderdale, Florida: J. Ross, 2009), 182.
3. McConnell, Steve. *Software Project Survival Guide*, (Redmond, Washington: Microsoft Press, 1998), 28.
4. Shore, James, and Shane Warden. *The Art of Agile Development*, (Sebastopol, California: O'Reilly Media, 2008), 383.
5. DeMarco, Tom, and Timothy Lister. *Peopleware: Productive Projects and Teams, Second Edition*, (New York: Dorset House, 1999), 6.
6. Poppendieck, Mary and Tom. *Implementing Lean Software Development: From Concept to Cash*, (Upper Saddle River, New Jersey: Addison-Wesley, 2007), 91.

Architech Solutions is a Toronto-based technology consulting and software development firm. We design and build powerful, user-centred systems that work. We're agile, disciplined, and passionate about delivering for our clients.

To book a free on-site Discovery Workshop led by our team of consultants, e-mail info@architech.ca or visit us at www.architech.ca



Architech Solutions
70 Bond Street
Suite 400
Toronto, Ontario
M5B 1X3

Phone: (416) 607-5618
Fax: (416) 352-1768

© 2011 Architech Solutions Consulting Services Inc. All rights reserved.

