Sizing Matters

Do not deny the classical approach, simply as a reaction, or you will have created another pattern and trapped yourself there. — Bruce Lee

Though agile methods are steadily gaining in popularity, not everyone has bought into them because they don't always see the benefits agile is supposed to bring to the table. Some of the frustrations include never ending development and lack of documentation. But the problem doesn't lie with agile because it was never intended to be a "silver bullet" solution. It was meant to offer a way to be adaptable to changing circumstances.

Unfortunately, many people have hidden behind the guise of "doing agile" development to cover up poor fundamentals, lazy behaviors, and a lack of mature and disciplined processes. No matter your chosen method of "how" to deliver, any successful software development effort cannot escape the importance of knowing and quantifying "what" you want to build. And to do this, you can't get away from certain elementary inputs to the estimation and planning process, one of which is size.

Don't be dogmatic

In 1967 world-renowned martial artist Bruce Lee realized that as proficient as he had become with the Wing Chun kung-fu style he studied for so many years, it was limited, as were all traditional fighting styles, when considered singularly. Doesn't a boxer with a longer reach have an advantage? What about the environment? Is a style that relies heavily on kicks, like Tai Kwan Do, as effective if you're fighting in closed quarters? One of Lee's students was basketball Hall of Famer Kareem Abdul-Jabbar who stood 7'2", while Lee was 5'7". Would a single style be equally effective for these two men?

Lee understood that these styles were developed with ideal situations in mind, but that didn't necessarily match reality, equating it to swimming on dry land. This led to his creation of a series of concepts, a style without style called Jeet Kune Do. But while Lee saw the value in not limiting his style and remaining fluid to his environment, he also saw the necessity to maintain key concepts, — which he called the "four ranges of combat" — that are fundamental to being a complete martial artist.

In 2001, a group of 17 software developers gathered together in a Utah resort to discuss the challenges and limitations of the traditional approaches to software development, recognizing the need to streamline the bloated, cumbersome processes that had been instituted over the years. Like Lee, they were looking for a way to reduce what is deemed unnecessary, but also realized that there were foundational areas or concepts that should remain. And here is where it seems many agile projects seem to miss the boat. Teams that believe that there is no need for foundational components of software development because they "are doing agile" (rather than "being agile") do so because they lack the discipline and process maturity necessary to implement the methodology appropriately.

The writers of the Agile Manifesto didn't say that process and tools weren't important, only that they believed there was greater value in individuals and interactions. It's not that following a plan isn't valuable, just that the ability to respond to change offered greater value than the dogmatic following of the plan. A mature development team, whether they prescribe to agile or not, understands this and will inherently demonstrate that level of discipline.

"The Agile movement is not anti-methodology, in fact many of us want to restore credibility to the word methodology. We want to restore a balance. We embrace modeling, but not in order to file some diagram in a dusty corporate repository. We embrace documentation, but not hundreds of pages of never-maintained and

rarely-used tomes. We plan, but recognize the limits of planning in a turbulent environment.” — Jim Highsmith

Is it bigger than a breadbox?

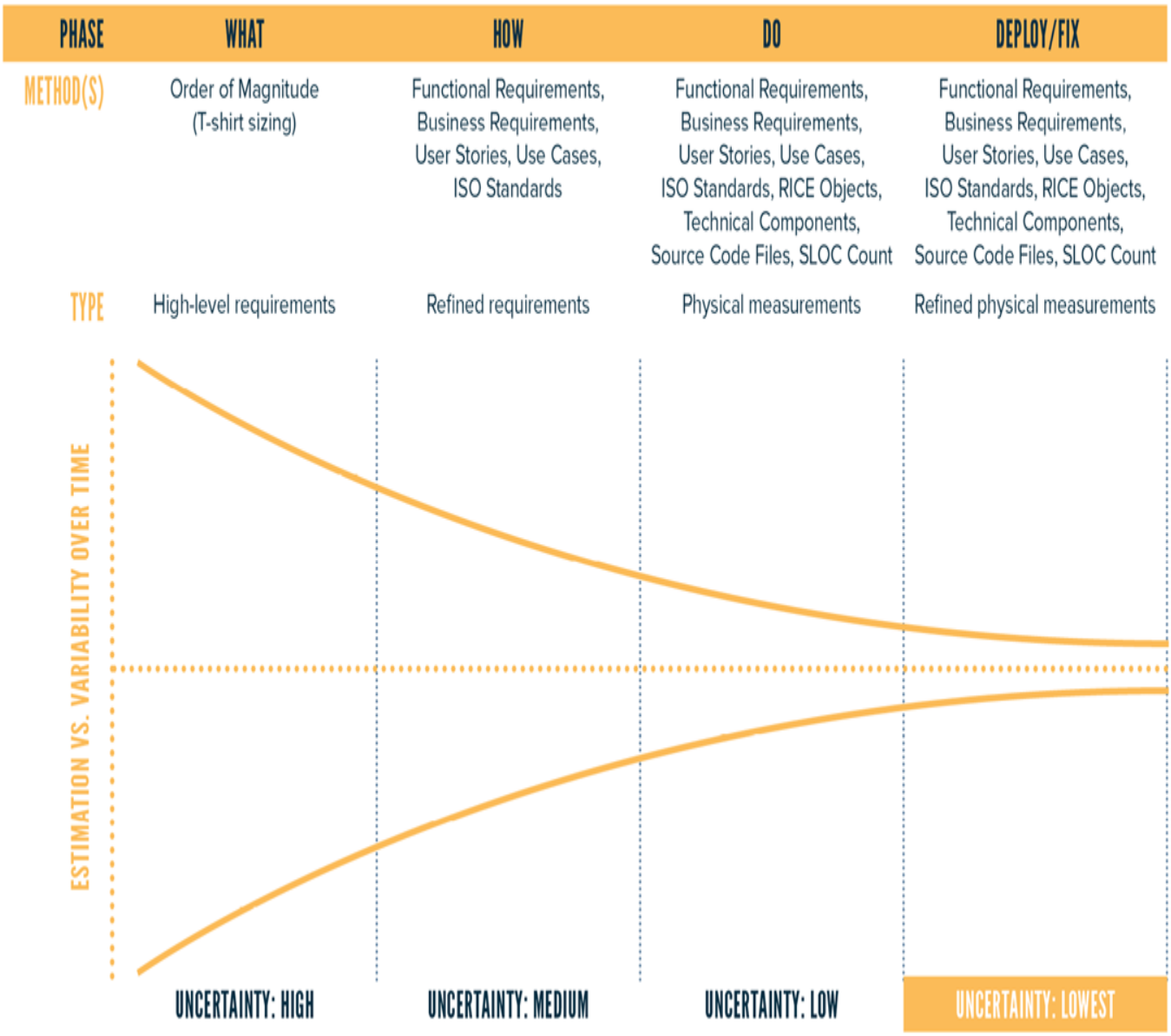
Agile development should be flexible enough to fit any container, but you need to know the size of the container. Knowing the size of the system you want to build is crucial to setting your team on a path toward success, however that is defined for you.

“But we’re doing agile, so we don’t gather requirements up front, thus can’t size the system.” That is nonsense. If I want a house built, I have to have an idea of how big it needs to be for my family to live in it and store my possessions. Why are we building this system? I must be able to find out some of the initial reasons behind the decision to build this software.

Will this be the final size? Most likely not, but that’s ok. Just like I can put on additions to my house, I can add to my software estimate. And this seems to be where some developers take issue with trying to size their efforts — *estimates are not commitments*. This is where the beauty of agile methodologies can really shine and show their true value, by providing a way to adapt to those changing needs. However, someone is footing the bill for the effort and should know what they are signing up for. Business value drives prioritization within a project (product backlog), so why shouldn’t it carry the same weight in deciding whether to do a project?

Where do I begin?

Agile says no upfront requirements should be gathered as those requirements will change so the details and specifics will be fleshed out as the project progresses. In the beginning all that is needed is an approximation that compares the relative size of the user stories identified. Planning poker is one popular exercise used to accomplish this. The same can be done with sizing. You only know what you know up front, but as time goes on greater details will reveal themselves about the system and the accuracy of your estimate will become more refined, as can be seen in below.



One of the 12 principles of the Agile Manifesto is that working software is the primary measure of progress. But progress can't be measured effectively without quantifying the size of the software that needs to be built, the size of working software completed to date, and the size of changes to scope. QSM's Sizing Infographic displays numerous ways to determine software size and correlations to project development. This offers an excellent starting point in determining just how big of a "house" your team is looking to build.

Fundamental elements, like sizing, remain crucial to software development projects. Embracing new delivery approaches, such as agile, does not mean we need to abandon practices that allow us to be successful in our delivery. The writers of the manifesto were looking to rid us of the layers of unnecessary routines that found their way into software development and get back to the basics.