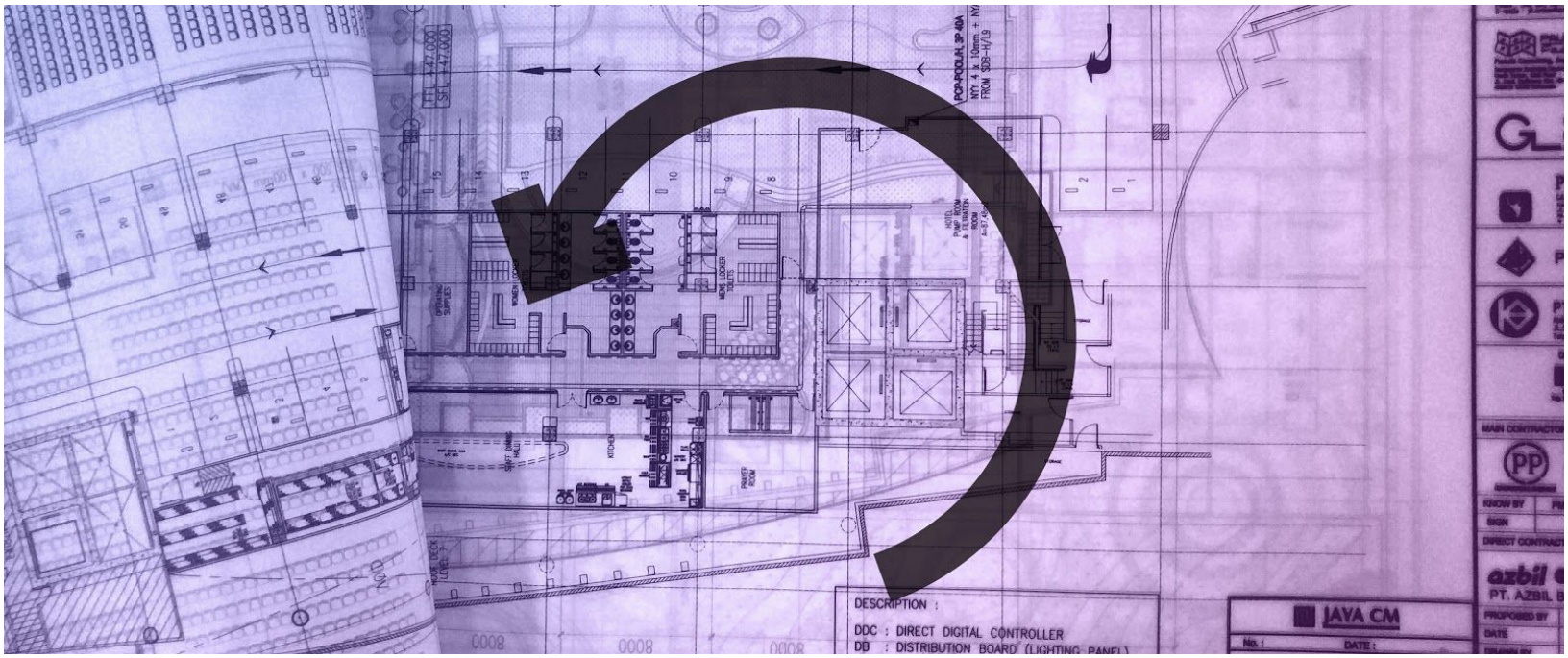
Agile Software Reboot

By John Kosh

What programmers should know and business stakeholders may not understand!



Agile by Default

Left to their own devices many professional software engineers working in small motivated teams isolated from administrative burdens tend to self organize if they are capable and self starters. This does not mean that all software engineers or developers are capable, self-starters, or even professionals. Fortunately however becoming capable and professional only requires a little self starting. Many software engineering professionals suffer through “Imposter Syndrome” when first wrapping their heads around Agile software methodologies and practices such as Scrum. Do yourself a favor and simply cut through all of the “cruft” and just learn the basic pillars.

So what is the meaning of Agile?

That all depends on your bounded context (perspective) and whether you are a pig or a chicken! The chicken and pig fable lends us some perspective of business stakeholders and software professional’s level of skin in the game.

A Pig and a Chicken are walking down the road.

The Chicken says: "Hey Pig, I was thinking we should open a restaurant!"

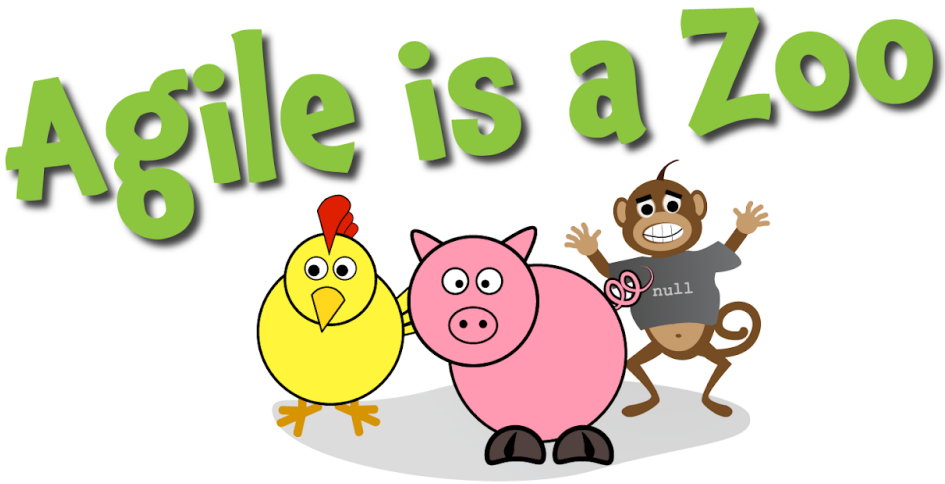
Pig replies: "Hm, maybe, what would we call it?"

The Chicken responds: "How about 'ham-n-eggs'?"

The Pig thinks for a moment and says: ***"No thanks. I'd be committed, but you'd only be involved."***

From the professional software engineering perspective the term Agile may have been coined to follow the more traditional concept and definition within the english lexicon of ***“quick and well-coordinated in movement”***. Potentially even the alternative of **“marked by an ability to think quickly; mentally acute or aware”** may also be quite applicable.

In either case the original intent of Agile was to provide a self-described statement of values and principles common among successful Agile software professionals. From its inception Agile principles were authored by software programmers, for programmers, to call out those important traits of successful software programming professionals.



When you forget the principles...

Where things went caddywhompus

On the other hand the chickens see Agile as something more akin to the newer definition of **"noting or relating to a philosophy of product development and production intended to create and distribute batches of working products in a short period of time with subsequent batches planned in a cyclical schedule of improvement, production, and distribution"**. Clearly the dictionary.com definition above exemplifies this corruption of state with the fact that it does not include anywhere a single mention of individuals, collaboration, or response to change. In fact it only highlights products, efficiency, distribution, and short cycles of delivery. Things critical only within the context of chickens egos and ambition to show their mastery of managing and administering processes for the sake of process management.

This is where the shiny lure and sexy lingo surrounding Agile terms and methodologies takes hold in the mind of business leadership. Since most of the technical concepts of programming are completely foreign to non-programmer professionals Agile terms and ceremonies are something simple they can grab hold of to administer and measure. Do the right thing and hire **only** expert talent that you will be able to trust they will perform and manage themselves effectively.

From 1942 to 1946 Major General Leslie Groves directed the U.S. Army Corps of Engineers in a top secret project. Major General Groves had very little scientific expertise but was talented in providing and managing resources and support for the world's top nuclear scientists who designed the atomic bomb. Major General Groves understood his role was not to dictate how the atomic bomb should be built, but to provide the direction of resources, leadership, and framework so that it could be built. This is the role of chickens.

Pigs are better than monkeys

Agile methodologies, practices, and ceremonies are sexy from the outside looking in, but the business stakeholders need to remember they are just spectators for a reason. Successful software development teams require competent leadership, talented engineers, and individuals that both ascribe to and practice professionalism in their craft. If you substitute code monkeys for pigs you are not only reducing the ability to create quality software, you are likely damaging quality, inducing rot, adding technical debt, and incurring opportunity costs and potentially damaging your competitive advantages within your industry.

It is not enough to hire a **code monkey** to sit in front of a computer 8-12 hours per day simply typing ubiquitous lines of code to add features or fixes described in a work item or task. Professional software engineering talent is required. The craft of engineering and architecting software is based on providing quality, extensible, working, testable, light-weight, secure, modular, and moldable understandable code without repetition to satisfy the needs of the business as a whole. This alone leads to shortened, predictable, and the flexible delivery of quality software. This is the role of Pigs.

Those organizations that fall trap to slick IT staffing firms producing mediocre cookie-cutter code or those managers who perceive Software Engineers and Programmers as assembly line workers and simple staff resources that can be thrown into Agile practices for "**cyclical schedule of improvement, production, and distribution"** are digging themselves into a hole of technical debt and software rot.

In 2020 Honda Corporation released a new more powerful, more reliable, fuel efficient, and higher torque lawn mower engine. This was accomplished by reducing the number of engine parts from 106 to 39. This exemplifies why quality engineering matters, and why Honda sells more lawnmowers than automobiles.

Being Agile

Today there are far too many purveyors of specific techniques and training programs to force you and your team into Agile practices and products. Many of these champions of Agile are not even professional software developers. Following too many of their rigid over-arching processes actually reduces your ability to be flexible, nimble, and agile in your approach to developing your team and software. Structure the agile world around you that makes sense and take what you learn about Agile practices with a grain of salt.

It has been nearly 2 decades since 17 programming experts sat down to craft a statement of what Agile is on agilemanifesto.org. Their hard fought experience and values still hold true as the best representation of what it takes to be agile in the craft of software engineering even today.

Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it.

Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

My take on the above items is…

Individual interactions are most effective in small groups with high degrees of participation. Face to face meetings with business stakeholders, team and group individual work/architecture sessions, pair programming, mob programming etc improve results. Use the most accessible and least complicated tool to do the task. Allow tools and processes only to exist as close to your code as possible, since expensive cumbersome tools with many gadgets add complexity and unnecessary processes and context switching.

Working software should always be critical. If you do not supply enough documentation within the code you write however you lose all ability to effectively update or maintain it thus, add only enough to ensure it is concise and able to be easily and quickly understood by you and others.

Maintaining documentation as close to code is the most efficient way to ensure living documents are continually up to date.

Customer collaboration requires both stakeholders and engineers to have small working sessions together in order to align concepts and linguistic translation of ideas. Isolating either side from each other leads to poor translation and feedback cycle effectiveness. It can also be dangerous to translate through multiple intermediaries (Stakeholder language -> Business Language -> Programmer language). Use design by example, spikes, proof of concepts when necessary to convey or vet your ideas.

Responding to change is more easily accomplished immediately when keeping development and feedback cycles short. Planning for future specific unknown changes or additions becomes exponentially difficult the farther you plan ahead with specifics rather than goals. Applying flexible and extensible architectural patterns enables you to quickly modify logic with fewer side-effects.

Principles behind the Agile Manifesto

We follow these principles:

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity--the art of maximizing the amount of work not done--is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Other Critical Points

To sum it all up... Impose high standards upon yourself and your team. Creating **quality** software is hard and extreme rigor must be imposed on eliminating external factors that damage the quality of software you produce. Be firm in your convictions, communicate, and collaborate as often as possible with individuals prior to estimating or agreeing to project plans. Always take the simplest, and correct approaches in designing software. Keep it DRY (Don't Repeat Yourself) and KISS (Keep it Simple Stupid). Commit code early and often, document effectively but not exhaustively, and test extensively. This used in conjunction with all of these Agile patterns will set you on a path to success.

Continuous Improvement

Approximately 90% of the experiences I leverage daily while writing software center around what not to do or what mistakes to not make. Always try to learn **what not to do** from those with holes in their feet. Defensive programming or even TDD if it suits your working style should be incorporated into your daily work. Integration and Unit tests are the only proof that your software works so learn to rely on them. If you through well considered design actually write less code you truly are providing better engineering solutions (consider the Honda lawn mower). Strive every day to improve your craft.

Stand on the Shoulders of Giants

Learn the **10 Commandments of Egoless Programming** as laid out in Gerald Weinberg's book "The Psychology of Computer Programming" - *circa 1971*. Live by the spirit of the **"Agile Manifesto"** and adopt the **"Twelve Principles of Agile Software"**. Take and adhere to the **"Scribes Oath"** (Programmers Oath) by Uncle Bob Martin. Learn Uncle Bob's **SOLID** patterns from his "Design Principles and Design Patterns" paper. Implement as often as possible the GoF (Gang of Four) patterns from **"Design Patterns: Elements of Reusable Object-Oriented Software"**.

Coup de Grâce

If you learn and take to heart the above, you should be well equipped in the required professional skills to be Agile even if you lack the expertise to be a Scrum Master, story point, or participate in sprint retrospectives. Those are just ceremonies and processes that cloud the intent of the Agile Software movement.