Agile & Architecting: Friends or Foes?

Erik Philippus ~ IMPROVEMENT BV
erik@agile-archtecting.com
version 1 - may 2009

The recent emergence of agile methods has shaken the foundation of the ivory tower architecture. A new breed of architects called Agile Architects is stepping up.

Architecture has always been controversial in the agile community. Should design be done up front, how much, and when does architecture turn into BDUF, the big bad wolf of agile programming? The agilists are right to some extend: most traditional architecture processes are heavy weight, long-winded and time-consuming. In circles of architects, however, agile methods are perceived as architecturally weak, disconnected from the realities of delivering large, multidisciplinary software-intensive systems in complex environments.

Agile Architecting is an attempt to combine the best of both worlds.

The Essence of Agile

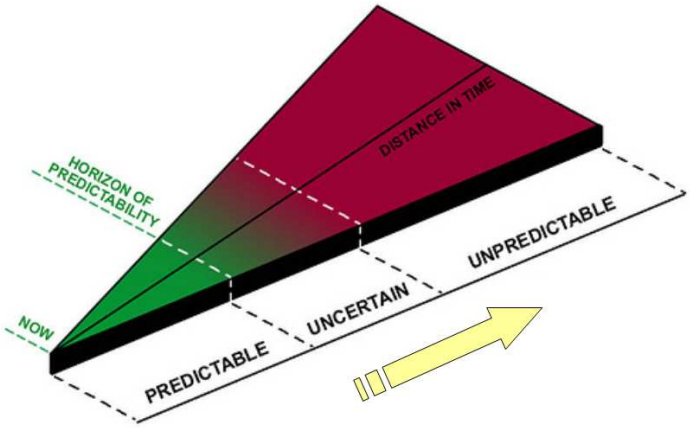
What does it mean to be 'agile'? In general terms, the phrase 'agile' refers to a person being very limber and dexterous, but also indicates a state of alertness and watchfulness. At first sight, this advanced interplay between body and mind sounds like a beneficial achievement for a yoga practitioner - so what exactly does it mean in the context of software development?

Agile is commonly described by its characteristics, such as iterative development, frequent inspection and adaptation, self-organizing and cross-functional teams, etc. Most of the agile methods are based on a set of core principles, focussed on software delivery and collaboration. The Agile Manifesto sums these up, by stating that it values:

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

These are merely the externally observable artefacts, but this palette of descriptive terms does not really answer the question: "what is the essence of agile?" Looking at the inappropriate application of the honorary title 'agile' to a wide range of practices that are in essence non-agile, there seems to be enough reason to elaborate upon the essence of the agile body of thought.

First of all, it is of vital importance to understand that the agile approach is a reaction to a software development tradition that goes back three decades. This tradition is rooted in the stubborn belief that predictability is a vital prerequisite for any industrial software development project. This doesn't really come as a surprise, given the prevailing post-war engineering paradigm of 'a makable world'. In view of the high percentage of failed software development projects, for many years the holy grail of software development was to gain full command of projects. Only when a project was executed according to a predefined masterplan, it would deliver what was specified at the start. The deeply rooted rational for this line of thinking is the seductive assumption that if you cannot predict a process, you cannot control it.



The connection between predictability and controllability is further intensified by the conviction that high perceived predictability is a protection against the stress of an imminent runaway project. As a result, we've seen a long procession of methods and tools which all promise that they would be able to send the escaped ghost of unpredictability back in the bottle.

Of course some useful engineering practices have surfaced the last decades, but the horizon of predictability of software projects has not been stretched significantly. In fact, the traditional approach has yielded a growing number of troubled or failed projects. As demonstrated by the rise and fall of the widespread waterfall approach, it is just not possible to fully command and control a sizeable software development project from sand to glass by using a serial lifecycle, starting from a BDUF (Big Design Up Front).

The siren song of the predictive approach is that it is based on a logical, linear model and thus "makes sense". Instead of questioning the paradigm, the methodology has been gradually expanded into a complicated, long-winded and sometimes even bureaucratic practice. We've been trying to create predictive requirements and communicate them perfectly up and down the hierarchy for over 30 years, and failed year after year. In most cases, the predicted delivery date is the first day of the schedule slip.

In Hebrew there's a saying: '*Tafasta Merube Lo Tafasta'*, which means: if you try to catch everything, you will catch nothing. In Lao Tzu's 'Tao Te Ching' a similar saying is found: 'Trying to control the future is like trying to take the master carpenter's place.

When you handle the master carpenter's tools, chances are that you'll cut your hand'. In other words: if you try to obtain perfect knowledge up front, it will have the reverse effect.

Agile accepts change, uncertainty (and therefore unpredictability) as the natural state. This can be a quite uncomfortable thought for most traditional managers, which is probably the reason why many organizations still value perceived predictability more than they value the ability to change direction quickly. Ironically, the agile approach gives you *more predictability* than those carefully planned-up-front project plans can provide!

The goal of an agile way of working is to promote collaboration, the interaction of people on interpersonal, cultural and structural levels. To refute a common misunderstanding: agile is not equivalent with 'quick'. Agility means adopting a posture that allows you to respond rapidly to changing market conditions and customer demands. In a nutshell: agile is adaptive instead of predictive.

Agile Architecting

Many agilists regard agile to be the antithesis of the project manager's favoritism toward plan-centric management. They sometimes caricature the role of the architect as an inhabitant of an ivory tower, surrounded by elaborated models and producing bulky architectural documentation that is subsequently assigned to the development team for realization. But in circles of architects, many agile methods are perceived as architecturally weak, disconnected from the realities of delivering large systems in complex environments.

Although it is not surprising that the traditional architect is regarded by most agilists as a typical representative of the old paradigm in software engineering, we must beware of throwing the child out with the bath-water. To dispose of architecture as the basis for the realization of sizeable and complicated software-intensive systems is short-sighted and would certainly move back the hands. On the other hand, the rise of the agile approach will inevitably force the traditional architecture community to reflect on a new interpretation of architecture. To some extend, the agilists are right: most traditional architecture processes are heavy weight, long-winded and time-consuming.

Agile architecting is an attempt to reshape the architecture process according to the new paradigm of system development, aiming at an increased level of agility in architects themselves, the architecture process, the resulting architectures, and delivered systems alike.

A beneficial side-effect of agile is that it highlights the inherent conflict of the architect. Dealing with perishable system requirements of imperceptible customers. Conflicts with the managers who want to see the project delivered yesterday instead of tomorrow. Daily struggles with developers who want to code now, and think later. Politically coloured trade-off decisions. As every seasoned architect knows, there are many such conflicts, some subtle, that keep the stress up and the spirits low. Agile architecting comprises all relevant and interrelated technical and non-technical skills that help the architects to respond effectively to these notorious dilemmas.

Architecting is the art of addressing all relevant cross-cutting concerns, balancing the needs of the different stakeholders. This remains fully applicable for agile architecting. The daily practice of an agile architect will still include articulating the architectural vision, conceptualizing architectural approaches, creating models, decompositions and interface specifications, and validating the proposed architecture against (quality) requirements and assumptions. The difference is in the level of responsiveness toward new (unexpected) information that becomes available during system development. The agile architect has an adaptive attitude that is opposite to the traditional belief that requirements and design solutions should be frozen as early as possible.

The agility of the architecture itself is another cornerstone of agile architecting. What is the distinguishing characteristic of an agile architecture? To remove a source of misunderstanding in advance: an agile architecture is not equivalent with a 'flexible architecture'. Although flexibility in itself can be a useful feature, the pursuit to arrive at an overall flexible architecture is like chasing the pot of gold at the end of the rainbow. In many cases, demanding flexibility as a generic architectural requirement rather points at indecisiveness. An agile architecture is based on a well-considered choice with respect to the variability of the different parts of the intended system. Some parts are explicitly modelled as members of an invariable and firm foundation, while other parts are deliberately designed as being highly flexible, or more precisely: scalable, configurable, portable, etc. The agile architect will make the subdivision, guided by a mixture of product/technical roadmaps, business considerations, market trends and fingerspitzengefüll.

The focus of the agile spectrum of methods is on dealing with uncertainties during development and manufacturing, based on the expectation that uncertainties related to ambiguities in customer requirements, the viability of new technologies, etc., can be resolved before the system is shipped. As we all know, this is an unrealistic proposition in most cases. In spite of that, the focal point of the agile approach has been on process innovation rather than product innovation. However, there are good business reasons to enhance agility not only in the development process, but also in the resulting system itself. Agile systems are characterized by demonstrable reusability, reconfigurability, portability and scalability. These quality attributes give systems the ability to change from one state or operating condition to another rapidly, without large switching costs or increase in system complexity. This is especially important for products that are long-lived, and require a significant upfront investment – the expense is simply too large for extensive renovation or even building an entirely new system each time the requirements change after initial fielding of the system.

In conclusion, the agile approach represents an exciting opportunity for architects to establish a streamlined and profitable development process for the realization of adaptable products based on an agile architecture.