Sustainable Architecture

Position paper for the workshop "Architecture is Dead – Long Live the Architect", OOPLSA 2002

Klaus Marquardt
Käthe-Kollwitz-Weg 14, D-23558 Lübeck, Germany
Email: marquardt@acm.org

Agile methodologies tend to be suspicious against software architecture. Namely XP appears to replace software architecture by the Metaphor and architectural spikes[Be]. The agile manifesto puts an emphasis on emergent architecture[Ag] that neither requires an architect's role nor specific tasks. What is so special and suspicious about software architecture, from an agile point of view? I see some (mis-)conceptions that are often implicitly assumed, though hardly expressed explicitly.

On the other hand, architects are reluctant in adopting large parts of the agile ideas. So what is special and suspicious about agile methodologies, from a software architect's point of view?

Prejudices about software architecture

- Architecture is an up-front activity that hinders in true agility.

  That is part of the truth, and a common view in some companies. However, few architects claim to have "nothing to do" after the inception phase of a project. They are mostly busy with helping the system to adopt to late changes and improved understanding of the problem it is supposed to solve.

- Architecture translates requirements into solutions.

  … which is impossible as the requirements are always potentially changing. In fact, this is a common misconception even among software architects. Architecture adds a lot to the requirements and sets up the technical maximes that can help implementing the solution. Most of the architecture can be done with knowing only a few top-level requirements, or even implicit assumptions.

- Architects are not developers.

  … and do not fit into the "travel light" paradigm. While some (especially large) companies make a clear distinction between architects and developers, this is not really common practice in software industry. A lot of successful organizations follow the "Architect also implements" pattern, and (depending on the team) architects can even loose their credibility if they fail to communicate their ideas via code.

- Architectures emerge from self-organizing teams.

  This may be the case occasionally. More frequently, a clear technical vision is a good guidance through the daily questions and problems. Such a vision becomes refined and revised, but it does not emerge – it takes dedicated time to think and let it grow.

- Structure is unrelated to function.

  In the end, only user visible functionality is worth the effort. Structure does not add any value – and is a typical focus of software architecture. Nevertheless, huge systems require some internal structure to be accessible for a team of developers. Industrial

projects are more sociological than technical events, and the technical setup needs to support the social issues. An appropriate structure is a big help here.

Prejudices about agile methodologies

- A big picture is not desired.

  It is desired, but agile approaches are reluctant to demand it very early. There is an initial idea that evolves over time, when the project's participants learn more about the project and its world. How large this initial idea needs to be, is a matter of discussion and needs to be answered within each project individually. Some agile projects have only been started after the system architecture was done, considered part of the external requirements.

- All developers are equal.

  … thus an architect will loose his/her status. Agility implies that the project can move swiftly in new directions, and that the people react to changes with a flexibility that could bring different tasks and roles. In the end, all necessary roles are highly valued, among them could be an architect. The interpersonal status of an architect has never really depended on a job title.

- Only functional code is valued.

  In the end, only user visible functionality is worth the effort. Structure, a concept typical to software architecture, is not valued by itself. However, everything that supports the visible function is desired and indirectly valued. An appropriate structure can be a big help for not-so-small projects. It gets harder to judge what is appropriate and what is not – decision about structure are based on the demand now.

Prejudices reflected

There are definitely more assumptions and prejudices. For the purpose of setting up the scene these are sufficient. Besides that different temperaments shine through, and the personality types of some architects may not fit with the personality types that agility requires, there is something else mixed up.

The agile prejudices against software architecture seem directed towards the product, the architecture itself – and make little distinction between the architect's role, the related process and activities, and the product, the architecture itself. So let's neglect the architecture itself for the moment, and focus on the related role.

Would it surprise you to hear that agile methodologies are often introduced and fostered by a team's software architect?[Ha]

The architect's role

The popular interpretations about what software architecture really is all about range from "a high level description of the system design" to "creating and sharing a vision". Apart from the architecture itself, this indicates two parts of the architect's role. The obvious part is a visionary, creative one; the more hidden part is supporting and influencing.

Both aspects of the role fit well into agile project – in combination. Each team, in particular a team following agile ideas, will be lucky to have a visionary person in its mid, provided that he/she can be integrated and shows a supportive attitude.

The question is: Are the two parts of the architect's role related, and how strong is that link?

The link between creative and supportive is the ability to provide ad hoc advice on any arising problem, the ability to answer questions in sufficient level of detail, and maintaining that the advice and answers are in sync with the vision. The driving desire of the architect is the will to bring a vision into reality; the strength of the link is given by the desire of the team to refer to the architect (not to the architecture[Wo]) and how open the architect is to these challenges.

The focus on the architect's role brings out the nice match between agile development and software architecture. So here is the answer to the observation: those architects with the largest influence already are eager to establish an agile attitude, because their influence might even grow. The teams with a supporting architect are probably more willing to adopt agile methodologies because it fits with the existing attitude and offers increased team and personal opportunities.

Sustainable architecture

In the above section, a miracle has happened: the architect was able to answer any question and to give good and consistent advice! This ability is related to experience, self-confidence and hard work. To help the miracle happen, the architect can rely on a number of techniques that are close to craftsmanship. Among others, the management of responsibilities and dependencies is essential, and the emphasis to update structure with each added or changed functionality.[FowPar]

The architecture that results can be called sustainable from several aspects. It does not consume more time or effort than it gives back to the project, not even in a short to medium time span. It does not drift apart from the project team's needs, it does not need to be more than one step ahead of development, and it can silently be adapted to changes in its environment, the project and its people.

However, the "architect" metaphor appears increasingly inappropriate for this kind of role. I like to complement it with metaphors from medicine. The architect of sustainable software needs a mindset that you also find with medical doctors: the attitude to "proactively wait".[Ma]

[Be] Kent Beck: Embrace Change. Extreme Programming Explained, 1999
[Ag] Online at: www.agilealliance.org
[Ha] Observation in the XP user group Hamburg, and of Neil Harrison (private communication)
[Wo] see also: the title of this workshop
[Fow] Martin Fowler: Refactoring, 1999
[Par] Parnas on aging code, keynote at ICSE 1994
[Ma] Dr. Kerstin Marquardt (private communication)