Editor: **Eoin Woods**
Endava
eoin.woods@endava.com

Aligning Architecture Work with Agile Teams

Eoin Woods

AGILE SOFTWARE DEVELOPMENT is a very commonly practiced approach for software delivery, and today the software architect's role in making key project design decisions is broadly recognized. However, in my experience, difficulties frequently arise when agile development teams and software architects work together. These difficulties—caused by differing priorities, languages, and, sometimes, development philosophies—often result in development teams dismissing software architecture as "big design up front" and, in turn, architects producing architectural work that isn't useful for agile teams.

> Certain architecture practices encourage architecture work that supports agile development.

Conflict between architects and agile teams is also problematic for organizations, causing them to miss the benefits that accrue when architects and agile development teams work effectively together. Agile teams are often great at delivering useful software in a timely manner, but I've seen them encounter problems with factors such as security, system monitoring, or systems integration, which product owners don't tend to prioritize. However, this is where software architects can help, because they address exactly these areas.

In an earlier column in this department, Eltjo Poort explained how the Risk- and Cost-Driven Architecture approach helps architects align their work with agile teams.[1] In this column, I propose some other, more general, ways to achieve this.

Agile Software Development

Agile software development can refer to many specific software development methods, the best known being Scrum and Extreme Programming (XP). What unifies agile approaches is their commitment to the well-known *agile manifesto*,[2] a philosophy that values

- software that works over comprehensive documentation,
- customer collaboration over contract negotiation,
- individuals and interactions over processes and tools, and
- response to change over following a plan.

Although the manifesto acknowledges that all these items have value, it elegantly makes the point that working software, customer collaboration, individuals and interactions, and responding to change are the most important.

Few people would disagree with these principles—who honestly prefers completed documentation over software that works? What's remarkable is the strong community that's formed around these simple statements to create agile software development.

Software Architecture

I've previously defined and characterized the software architect's role,[3] but I'll summarize it again here: architects are responsible for the design decisions that are risky, costly, hard to change later, or all three. The architect also

- focuses on design work;
- meets the needs of a wide stakeholder community (well beyond users and acquirers);
- addresses systemwide concerns (which are often nonfunctional);

0740-7459/15/$31.00 © 2015 IEEE

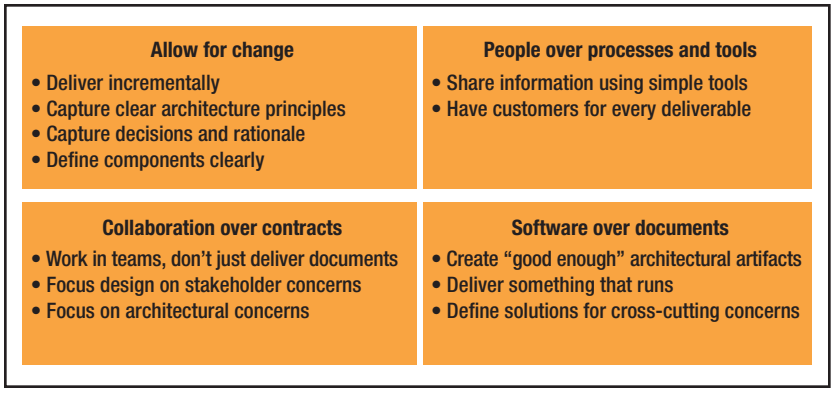

FIGURE 1. Practices for successful agile architecture.

- balances competing concerns to find acceptable solutions to design problems; and
- provides the leadership required to ensure that the system's architecture is well understood and supports its successful implementation.

Working Together

Having worked as a software architect with many agile teams over the last 10 years, I've realized that certain architecture practices encourage architecture work that supports agile development. These practices were inspired by and draw on different aspects of the agile manifesto (see Figure 1).

Let's briefly explore each practice to understand how they fit together.

Allow for Change

The agile manifesto urges architects to embrace change rather than rely on a plan. The following practices make architecture work more amenable to change.

Deliver incrementally. When working on a significant piece of design work, it's tempting to seek perfection before sharing it. However, agile principles encourage the opposite: architects should initially make a minimum number of key decisions and then a flow of decisions when needed to meet the system's delivery goals. This approach allows the work to respond to changing priorities and the increased knowledge that's available as the project progresses. In practice, I've found that initially you need just enough architecture work to define the key system structures, address the main risks, and get the work started. You can then refine the architecture as the project develops. Eltjo Poort explained how to use risk and cost to drive this decision flow.[1]

Capture clear architecture principles. Agile teams need each team member to be able to make good design decisions. Architects can support this by defining clear principles that help team members understand why the architectural structures exist and the architecture's most important characteristics. Such principles give team members a mental framework they can use when extending and changing the architecture.

Capture decisions and rationale. In a similar way, architects can help the team understand decisions by capturing important design decisions and rationale[4] and verbally explaining them regularly so that they become part of the project's aural tradition.

Define components clearly. As systems evolve, new components are introduced, existing ones are changed, and component interactions are altered. Architects need to clearly define each component: each one needs a good name, a clear set of responsibilities, well-defined communication interfaces, and a precise set of required dependencies. Without this knowledge, it's very difficult for the system structure to evolve coherently.

People over Processes and Tools

The agile manifesto reminds software architects that people, not processes and tools, create the software. The following key practices reflect this.

Share information using simple tools. A key part of a software architect's job is communicating the architecture to relevant stakeholders, such as infrastructure designers, end users, and compliance officers, as well as the development team. Although sophisticated modeling tools can be valuable, particularly for the software architect, I've found that effective stakeholder communication is best achieved through simple, familiar mechanisms such as wikis; presentations; and short, accessible documents.

Have customers for every deliverable. It's easy to write a seemingly important document before considering who should read it. This approach often results in a document that no one wants. To focus the architecture work on high-value areas, architects should have a defined purpose and audience in mind when cre-

ating deliverables, thereby maximizing their usefulness.

Software over Documents

Agile software development emphasizes the value of software that works over documentation describing how it should work. This principle has inspired the following architecture practices.

Create "good enough" architectural artifacts. Because a system's architectural models and documents can be quite significant pieces of work, architects take understandable pride in creating comprehensive, polished results. However, these deliverables aren't part of the final system and should be fit-for-purpose rather than polished to perfection. This isn't an excuse for sloppy work, but rather a guide for when it's time to move on to the next task.

Deliver something that runs. Everything we do needs to be validated and tested, whether it's production code or design principles and ideas. In architectural design, this means that rather than creating documents full of theoretical ideas, architects should deliver something that validates the architectural thinking. The best way to do this, in many cases, is by developing examples and prototype implementations that validate the key ideas, prove their feasibility, and allow them to be understood and adopted quickly.

Define solutions for cross-cutting concerns. Some design aspects are complicated because they affect many aspects of the system's implementation (they "cut across" the system's structure). These design decisions normally address qualities like security or scalability and should be a major focus of the architecture work. Delivering clear, proven solutions for each of the system's important quality properties is tremendously valuable because it frees the team to focus on design aspects that might be more interesting to the product owners, while still allowing them to achieve the required qualities in the system.

Collaboration over Contracts

The final aspect of the agile manifesto urges software architects to collaborate rather than maintain formal boundaries and agreements. Collaboration is key to effective architecture work and is supported by the following practices.

Work in teams, don't just deliver documents. A common complaint about software architects is that they work separately from the people affected by their decisions. To avoid this, architects need to collaborate with development teams, working directly with and in teams wherever possible. This allows ideas to flow both ways—helping the architect understand the real problems to be solved and helping the team use and develop the architecture as the system is built.

Focus design work on stakeholder concerns. Software architects can easily create architectures that are "scalable" or "flexible" or "efficient" without considering what the system's stakeholders really need. Therefore, another aspect of collaboration is ensuring that the architectural design work aligns with stakeholder needs and focuses on the system's most important aspects.

Focus on architectural concerns. When working with development teams, software architects must be engaged in a way that teams find valuable. I've found this is best achieved by focusing on the cross-cutting concerns in the system, such as common design patterns, how the system will be deployed, and how the critical qualities (such as security or availability) will be achieved. You might remember the "architecting in the gaps" metaphor for identifying architecture work.[5] Taking ownership of these "gaps" is a useful service that architects can provide to the team.

Software architects and agile development teams have a mixed history of working together, which is unfortunate but rectifiable. With a little flexibility and goodwill on both sides, software architects can channel their efforts through practices aligned with agile development's underlying principles. By using these practices, architects can work constructively with agile teams and significantly contribute to a project's success.

References

1. E.R. Poort, "Driving Agile Architecting with Cost and Risk," *IEEE Software*, vol. 31, no. 5, 2014, pp. 20–23.
2. "Manifesto for Agile Software Development," 2001; www.agilemanifesto.org.
3. E. Woods, "Return of the Pragmatic Architect," *IEEE Software*, vol. 31, no. 3, 2014, pp. 10–13.
4. P. Kruchten, R. Capilla, and J.C. Dueñas, "The Decision View's Role in Software Architecture Practice," *IEEE Software*, vol. 26, no. 2, 2009, pp. 36–42.
5. E. Woods, "Architecting in the Gaps: A Metaphor for Architecture Work," *IEEE Software*, vol. 32, no. 4, 2015, pp. 33–35.

EOIN WOODS is the chief technology officer at Endava, a European IT services company. Contact him at eoin.woods@endava.com.