The conflict between agile and architecture

Myth or reality?

Simon Brown
I help software teams understand software architecture, technical leadership and the balance with agility

(I code too)
What is agile?
What is **architecture**?

**Structure**
- The definition of something in terms of its components and interactions

**Vision**
- The process of architecting, making (significant) design decisions, etc
The conflict between agile and architecture

Myth or reality?
Myth

There is no conflict between agile and architecture

All software projects need structure and vision
1. A conflict in team structure

**Dedicated software architect**

Single point of responsibility for the technical aspects of the software project

**Everybody is a software architect**

Joint responsibility for the technical aspects of the software project
I’m a software architect
Software development is not a relay sport.
Architects?
We don't need no stinkin’ architects!
Small teams of generalising specialists, everybody does everything

With agile, there is often a perception that you must have self-organising teams
2. A conflict in process

Big up front design
Requirements capture, analysis and design complete before coding starts

Evolutionary architecture
The architecture evolves secondary to the value created by early regular releases of working software
The conflict relates to the desired approach

Moving fast, embracing change, delivering value early, getting feedback vs Understanding everything up front, defining a blueprint for the team to " follow"
Responding to changeover following a plan

This doesn't mean "don't do any planning"!

Manifesto for Agile Software Development, 2001
Modern software development teams often seem afraid of doing analysis.
No
design up front
Agile software team

We don’t need software architecture;

we do TDD
The result of the conflicts?
Chaos!

Does the team understand what they are building and how they are building it?
Chaos!

Does the team understand what they are building and how they are building it?

No defined structure, inconsistent approaches, big ball of mud, spaghetti code, ...

STOP

Slow, insecure, unstable, unmaintainable, hard to deploy, hard to change, over time, over budget, ...

Chaos!
Shared vision of WTF?!
Software architecture in the 21st century
Does the team understand what they are building and how they are building it?

No defined structure, inconsistent approaches, big ball of mud, spaghetti code, ...

Stop

Slow, insecure, unstable, unmaintainable, hard to deploy, hard to change, over time, over budget, ...

Let’s agree on some things

Let’s make the implicit, explicit

Put some boundaries and guidelines in place
1. A conflict in team structure

- Dedicated software architect
  - Single point of responsibility for the technical aspects of the software project

- Everybody is a software architect
  - Joint responsibility for the technical aspects of the software project
Every software development team needs a master builder.
Depth
Deep hands-on technology skills and knowledge

Good software architects are master-builders

Generalising

Breadth
Broad knowledge of patterns, designs, approaches, technologies, non-functional requirements ...

Awareness of options and trade-offs
The software architecture role

Dedicated software architect
Single point of responsibility for the technical aspects of the software project

Everybody is a software architect
Joint responsibility for the technical aspects of the software project

From chaos to self-organising

Elastic Leadership (Roy Osherove)
Chaos (command and control), learning (coaching), self-organising (facilitation)
2. A conflict in process

**Big up front design**
Requirements capture, analysis and design complete before coding starts

**Evolutionary architecture**
The architecture evolves secondary to the value created by early regular releases of working software
How much **up front design** should you do?

Big design up front?

Emergent design?
(or none, depending on your viewpoint!)

Waterfall

RUP®
Rational Unified Process®

XP
Extreme Programming

Atern

Scrum Alliance®
transforming the world of work

Something in between?
You should do

“just enough”

Isn’t agile about being flexible and adapting to the context? :-)}
What’s important?

Significant decisions

- Understanding the significant elements and how they fit together
- Understanding how security will work

Low-level details

- Class and sequence diagrams covering every user story
- Defining the length of all database columns
Just enough up front design to understand the structure of the software and create a shared vision for the team.
You need to identify and mitigate your highest priority risks.

Things that will cause your project to fail or you to be fired!

Agile software projects do have risks, right? :-}
<table>
<thead>
<tr>
<th></th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Low (1)</td>
</tr>
<tr>
<td>Low (1)</td>
<td>1</td>
</tr>
<tr>
<td>Medium (2)</td>
<td>2</td>
</tr>
<tr>
<td>High (3)</td>
<td>3</td>
</tr>
</tbody>
</table>
Risk-storming

A collaborative and visual technique for identifying risk
The role

The process

“just enough”
software architecture

Understand how the significant elements fit together

Identify and mitigate the key risks

Provide firm foundations and a vision to move forward

Software Architecture Document
Is a collaborative and lightweight approach to software architecture the missing piece of the jigsaw?
Do whatever works for you

simon.brown@codingthearchitecture.com
@simonbrown on Twitter
Buy the ebook for only $10

1GfwZBLaUKAM (code expires 8th May 2013)