Agile Software Architecture
how much is enough?

JAX London
April 2011

Eoin Woods
www.eoinwoods.info

About Me

- Software architect at UBS Investment Bank
  - responsible for synthetic equity platform in Prime Services
- Software architect for ~10 years
- Author of "Software Systems Architecture" book with Nick Rozanski
- IASA and BCS Fellow, IET member, CEng

Defining Software Architecture

Defining Software Architecture

- A common definition …

> *The software architecture of a program or computing system is the **structure or structures** of the system, which comprise software **elements** the externally visible **qualities** of those elements, and the **relationships** among them*
>
> Len Bass, Paul Clements and Rick Kazman (SEI)
> Software Architecture in Practice, 2nd Edition

Defining Software Architecture

- An alternative definition …

> *The set of design decisions which, if made wrongly, causes your project to be cancelled*
>
> Eoin Woods
> www.sei.cmu.edu/architecture/definitions.html

Essence of Software Architecture

- It is a design based activity
  - designing something (a system, a process, …) is key
- It requires focus on stakeholders & their concerns
  - you serve a wide constituency
  - you often need to clarify poorly defined problems
  - your work allows identification of risks and opportunities
- It addresses system-wide concerns
  - e.g. qualities rather than detailed functions
- It involves balancing competing concerns
  - no right answer / least worst option
- It includes providing leadership

Agile Principles and Software Architecture

The Agile Manifesto Values

More Value	Less Value
Working software	Comprehensive documentation
Customer collaboration	Contract negotiation
Individuals & interactions	Processes & tools
Responding to change	Following a plan

Architects and agile teams share many priorities

- Focus on the consumers of the systems
- Efficient delivery of valuable software
- Simplification and reduction of cost
- Quality & reliability of delivered software
- Supporting efficient change
- Effectiveness of communication

Reasons for Architecture and Agile Disagreement

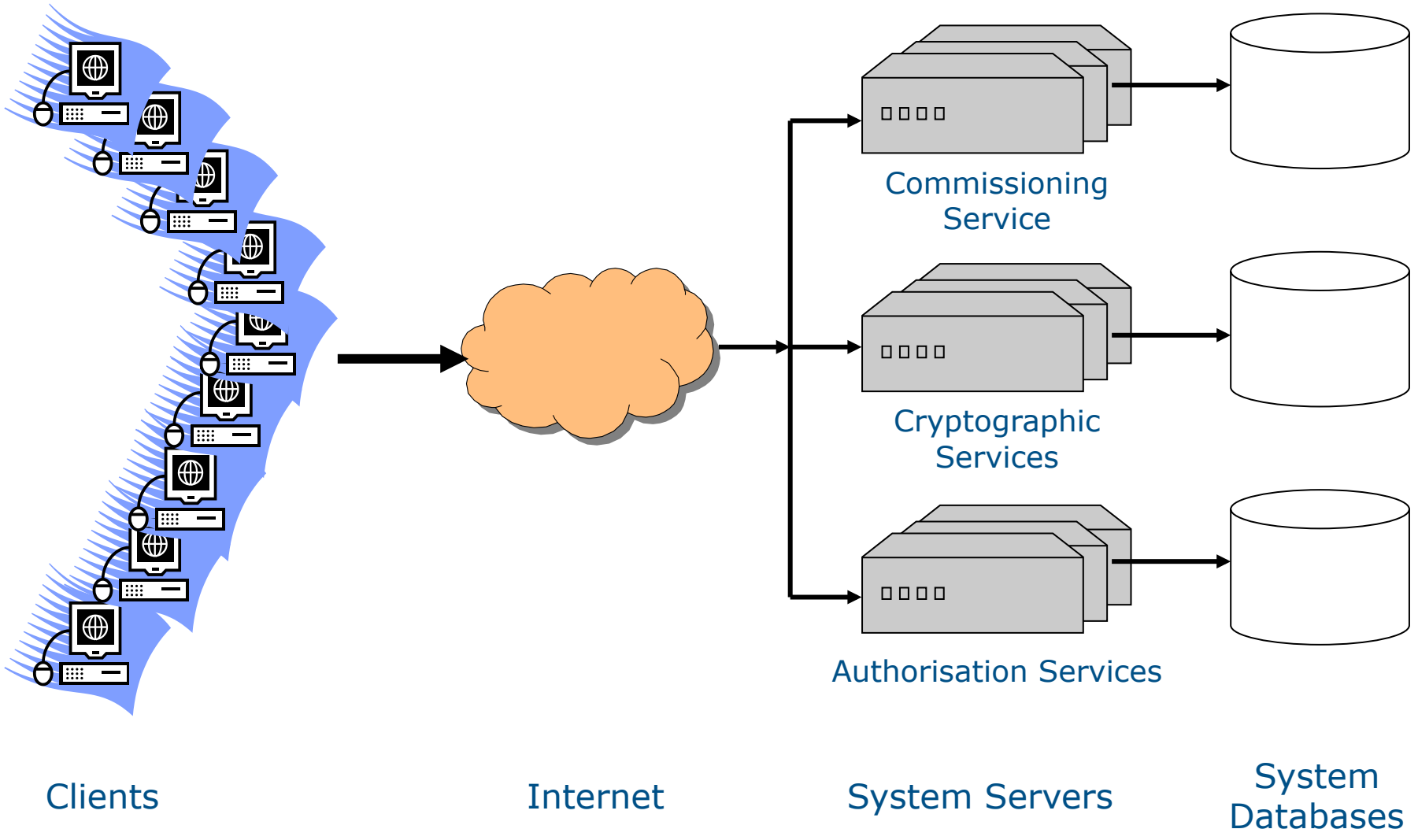
- "Big Up Front Design"
  - particularly when architects produce large documents
- Document centric vs. delivery centric
- Decision making vs. delivery responsibility
- Different views on processes and controls
- Differing time horizons for return on investment
- Architectural vs. "customer" (user) priorities
  - more customers than just the end users
- Agile deliveries in larger change programmes

Case Study

Case Study: Why mix architecture and agile?

- A small company needed to create a new generation of its product very quickly (to market in 9 months)
- The architecture needed 5 new servers from scratch
- The product must support Internet scale loads
- The previous product had a bad reputation for reliability, supportability, scalability and performance
  - this had to be sorted out this time around
- Server engineers were 400 miles away from the rest of the engineering team (and the server architect)
  - and they'd just been acquired from another company!

Case Study: A New Server Family



Case Study: Could an "agile" approach help?

- The team was relatively small (~10 developers)
- The team had domain knowledge already
  - all previously worked on similar systems prior to acquisition
- Range of experience all at least 18 months
- A product manager was in place
  - product development's answer to the "on site customer"
- Senior management didn't mandate process
- But where did "architecture" fit?
  - major refactoring after 1.0 wasn't an option
  - needed to get system qualities right first time
  - needed to move quickly as a coordinated unit

Case Study: Key Risks

- Tight timescales result in "just to do it" approach
  - danger of fragmented approach => unsupportable product.
- No time for a lot of BUFD and risk mitigation
  - couldn't build a framework to force commonality
  - time for small prototypes rather than real PoCs
- Focusing on some requirements could cause others to be neglected
  - e.g. performance vs. scalability or operational control
- Delivery focus could result in maintenance & operational difficulties
- Geographical split made communication difficult
  - new to the company so no prior relationships to fall back on

Case Study: Architecture practices (1)

- Define components clearly
  - prevented overlap or confusion within the architecture
  - helped to prevent scope creep
- Define clear design principles &
  Capture design decisions and rationale
  - made sure the key decisions were agreed up front
  - only bothered with principles that really mattered
- Good enough models and documents
  - allowed a sharp focus on what was actually needed

Case Study: Architecture practices (2)

- Define solutions for cross-cutting concerns
  - made sure that security and HA in particular were standard
  - saved a lot of redundant design time across the servers
- Deliver working examples and prototypes
  - a demonstration server was created with the lead developer
  - beyond this a production server was delivered!
- Have customers for all deliverables
  - no time for anything that wasn't valuable
  - deliverables that didn't get used / read / reviewed / modified were formally abandoned

Case Study: Architecture practices (3)

- Work in the teams
  - the server architect was embedded in the server team
  - weekly travel between the sites, daily conference calls
  - working on the production code as well as architecture
  - tremendous help in building mutual respect and relationships
  - very valuable feedback into the architectural design
- Address Real Stakeholder Concerns
  - little time to delivery anything beyond the essentials
  - every feature & quality was challenged for value
  - the critical requirements dictated the delivery timeboxes

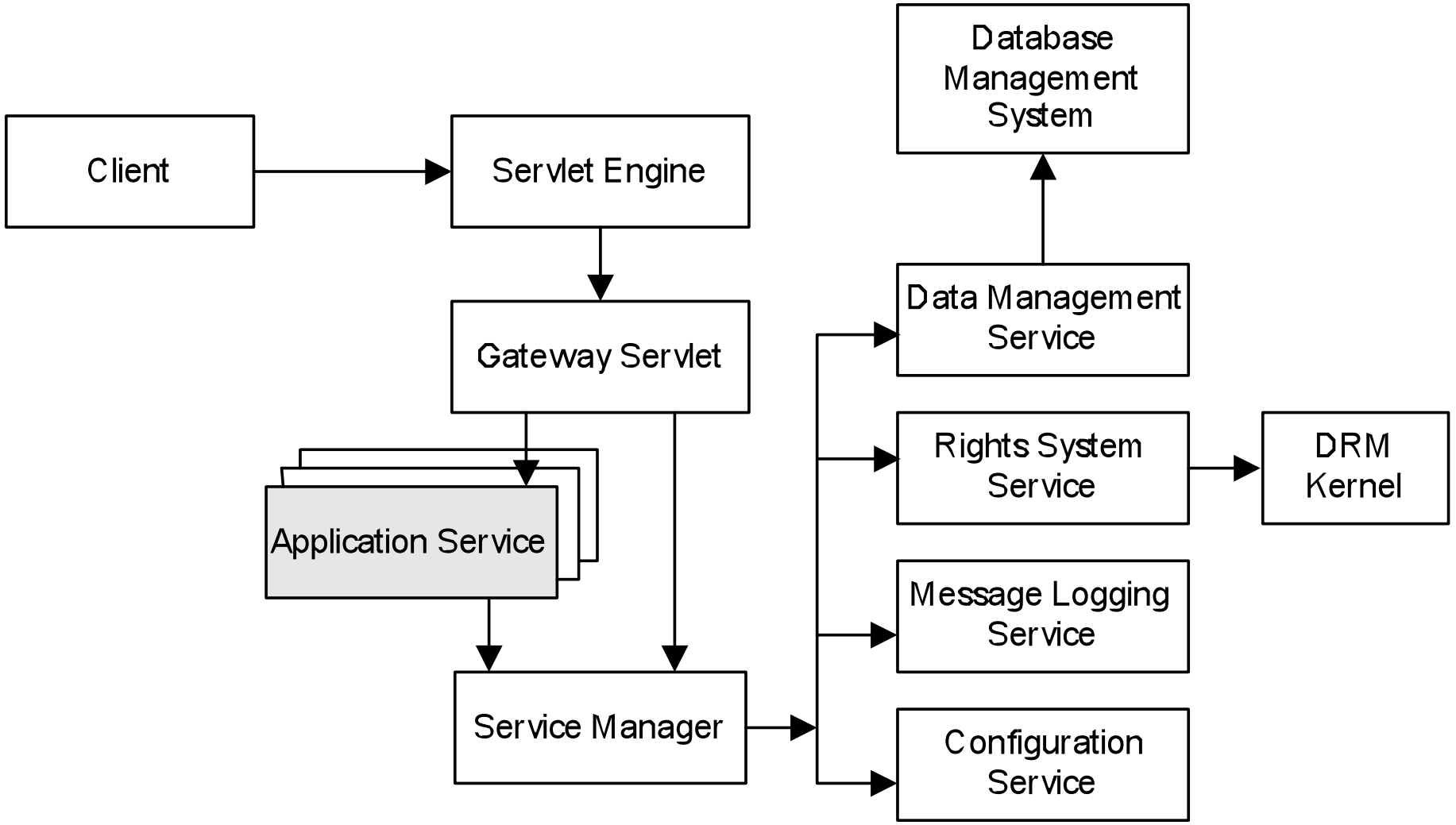
Aside: How to Work With Teams

- Solve their problems
  - have proven solutions for integration, security, DR, …
  - immediate value to development teams
- Jointly develop your architectural principles
  - ensure they are understood and agreed
- Collaborate rather than police
  - review to share & improve, not to govern
  - teams own their own work and should have pride in it
- Stay out of internal decisions
  - unless invited or you need to avert catastrophe
  - collaboration will mean that you have input anyway
  - when invited, you know you've succeeded

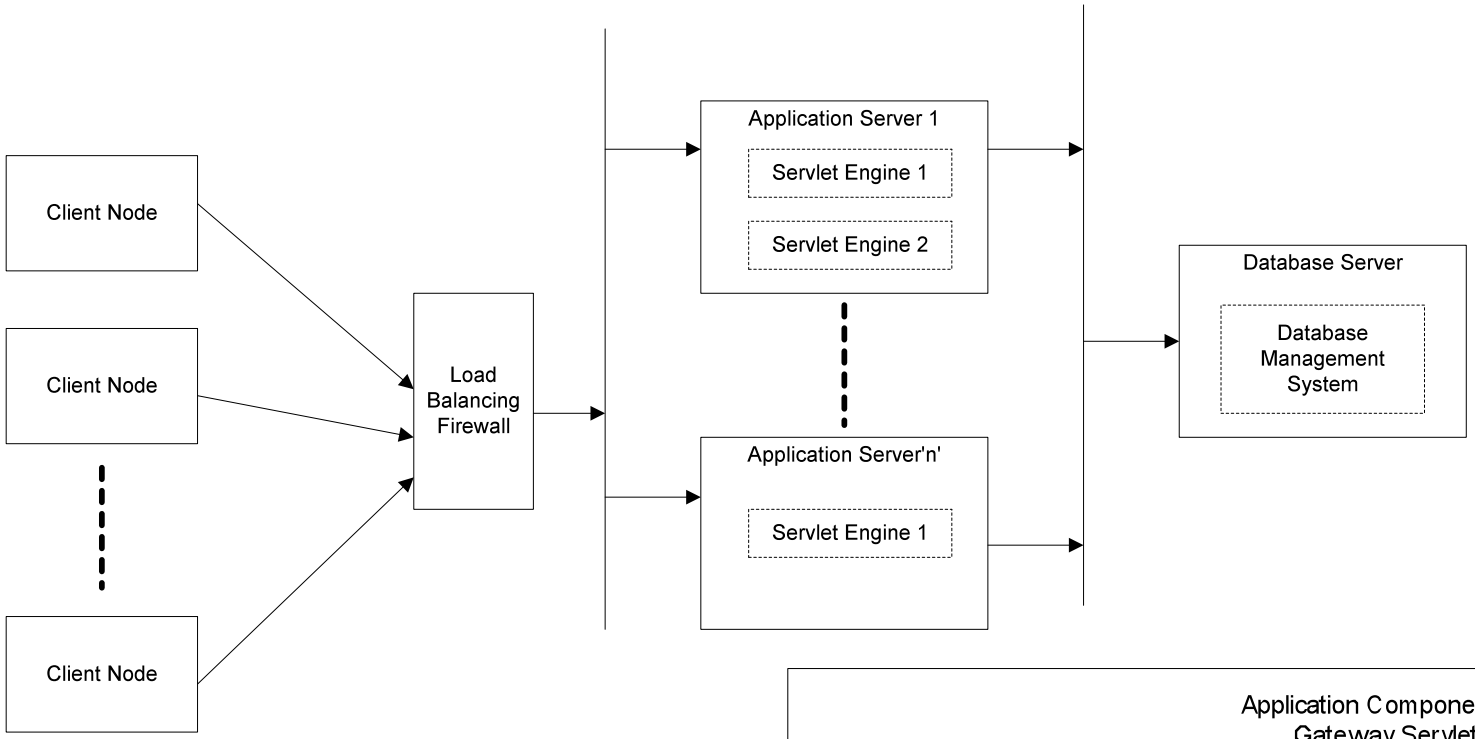
Case Study: Architecture Deliverables

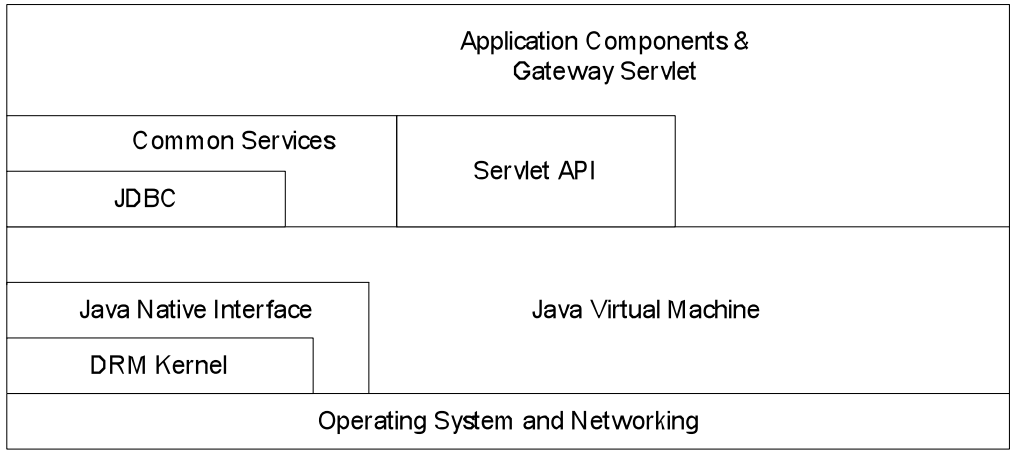
- A server product line architecture
  - 4+1 views to define the essential commonality with rationale
  - defined standard technology
  - defined cross cutting mechanisms and conventions
  - defined a standard design pattern for the servers
- Development standards
  - essentially the "development view" of the architecture
  - defined i18n, logging, request handling, technology to use, test approach, exception handling, product configuration, …
  - Anything not here or in the architecture was up for grabs
- The implementation of one of the servers
  - architect suffered his own decisions
  - resulted in rapid refinement of the product line architecture!

Case Study: Functional View



Case Study: Deployment and Development Views





Aside: Good Enough Modelling

- What is good enough?
  - consider currency, precision, detail, completeness
- Experience suggests
  - focus at the component/interface/connection level
  - prefer models & databases to pictures
  - be precise, even when abstract
  - ensure models can be updated easily
  - model with a purpose (audience and use)
- Areas to consider
  - functional structure, deployment, information

Case Study: Practices we missed

- Deliver work incrementally
  - the server software was developed incrementally
  - the architect was largely done up front
  - timescales meant backtracking was a problem
  - some mitigation using the "demonstration server"
- Share information simply
  - some use of Wikis, but mostly documents and word of mouth
  - could have greatly improved this with longer timescales
- Focus on architectural concerns
  - architect got bogged down in implementation detail
  - sometimes lost focus on architectural concerns

Case Study: The Results

- Ultimately successful because product was delivered
  - first version ready for beta testing within 9 months
  - deployed at major mobile manufacturer test labs in this time
  - judged to have met all of its significant requirements
- The architecture was definitely used
  - the server developers all used it
  - high degree of commonality achieved (without much reuse)
  - parts of it were refined right through the development cycle
- Server team became much more integrated
  - challenge of problem forced communication
  - architecture provided context for decisions and discussions
  - something to "rally around" when dealing with other teams
  - architect working in the team had a lot of beneficial results

The Twelve Architect's Practices

Allow for Change	• Deliver work incrementally • Define clear design principles • Capture design decisions & rationale • Define components clearly
Delivery over Documents	• Create "good enough" models & documents • Define solutions for cross-cutting concerns • Deliver working examples and prototypes
People over Processes	• Have customers for all deliverables • Share information simply
Collaboration over Contracts	• Work in the teams • Focus on architectural concerns • Address real stakeholder concerns

Conclusions

How do you work with an agile team?

- Start with the smallest amount of architecture possible
- Look for the risks that the project is facing
  - particularly around long term concerns and quality properties
- Where can architectural design specifically help?
  - avoid the problems where the team will help itself
- Apply architecture practices in a sympathetic way
  - it's often as much about presentation as substance
  - align with how the team is working
- If you can't find any more benefit which can be "sold" to the team and the customer, then you have "enough"

Signs that sufficient architecture is present

- Everyone can explain the overall system structure
- Engineers understand their modules
  - responsibilities, interfaces, interactions
  - impact of the module on the larger whole
- A credible argument can be made that you can provide all of the important qualities
  - performance, scalability, security, administration, ...
- There is general agreement on standards and norms
  - common "look" to code, same tools in general use, reuse of patterns and software, documentation testing approach, ...
- People know how things works and how we do things

Summary

- There is no need for agile vs. architecture conflicts
  - both schools of thought want to achieve the same thing
  - a little less zeal and a little more reflection on both sides
- The onus is generally on the architect to fit in
  - though the team need to make it possible
- The problem is often presentation not substance
  - this isn't about new practices, more tailoring proven ones
- There are practices that can address each part of the agile manifesto
- An architect should plug the gaps the development team is facing, not try to control everything

Questions and Comments?

Eoin Woods
www.eoinwoods.info
contact@eoinwoods.info