ARCHITECTING IN THE GAPS

Eoin Woods
www.eoinwoods.info
About Me

- Software architect & development manager at UBS Investment Bank
  - working on equity swaps systems in Equity Derivatives
- Software architect for ~10 years
- Author of “Software Systems Architecture” book with Nick Rozanski
  - new 2nd Edition published in October 2011
- IASA and BCS Fellow, IET member, CEng
Agenda

- Difficulties with Practicing Software Architecture
- Architecture is in the Gaps
- Exploring the Metaphor
- System Qualities and Boundaries
Difficulties with Practicing Software Architecture
Identifying Software Architecture

• software architecture = \{elements, form, rationale\}
  • Perry and Wolf (1992)

• The software architecture of a system is the set of structures needed to reason about the system, which comprise software elements, relations among them, and properties of both
  • Bass, Clements and Kazman (2011)

• The set of design decisions which, if made incorrectly, will cause your project to be cancelled
  • Eoin Woods

Yet, it’s still hard to be clear where architecture starts and stops
Key Characteristics of Architecture Work

- Design centric activity
  - designing something is key

- Stakeholder focus
  - wide community with conflicting needs

- System-wide concerns
  - architectural design decisions affect system-wide qualities

- Balancing of concerns
  - no right answer

- Leadership
  - responsibility for the work of others as well as your own
Software architecture is difficult to tie down
  • we know what design is
  • we know what implementation is
  • we sort of know architecture when we see it

But does this matter?
  • yes and no
  • life can be harder when we’re not quite sure

If you don't know where you are going, you will probably end up somewhere else
  • Lawrence J. Peter (of “The Peter Principle” fame)
Difficulties in Practice

• Justifying Software Architecture
  • do we really need it? what does it offer?
  • don’t we just do this stuff all the time?

• Knowing How Much Is Enough
  • how much of your time is “architecture”?
  • when can we get started on the “real” work?

• Where to Focus
  • what do you need to do that others won’t do?

• How to Work with Others
  • how do you cooperate with implementation teams?
  • how do you get people to listen to you?!
We Need a Good Metaphor

• When creating XP Kent Beck realised the need for a unifying idea to guide design work
  • the “metaphor” in Extreme Programming

• Somehow we need a metaphor to guide people’s thinking about software architecture
  • something simple
  • something that can be visualised
Architecture is in the Gaps
Software Architecture is found at Boundaries

User Interface → Services → Database

Services → Calculation Engine

Services → Change Notifier

Batch Processor
Software Architecture is at the Boundaries

- Architecture organises, links, unifies and constrains
  - shared ideas, shared interfaces, shared concepts, ...
  - Software architecture itself isn’t the end, it’s the means

- Software architecture allows others to work effectively and collaborate
  - even when they are unaware of this
  - provides the structure in which others can place their work
  - allows cooperation by providing unifying ideas

- Software architecture aims to achieve cross-element systemic qualities (“quality properties”)
  - qualities rely on structures, connectors and constraints
Examples of Architecture at Boundaries

- **Technical:** middleware and EAI
  - architecture as software
- **Organisational:** cross-team (“domain”) architect
  - architecture as coordination
- **Conceptual:** reference models for domains
  - architecture as a unifying metaphor
- **Design:** system components and connectors
  - architecture as context
- **Qualities at Boundaries:** security or availability
  - qualities are only achieved by unifying across components
Exploring the Metaphor
(or "so what?")
Where the Metaphor Can Help

- Identify where architecture is needed
  - in the gaps where no one else is looking
- Define architecture’s specific contribution
  - thinking about boundaries that no one else is looking at
- Distinguish the architecture work
  - the boundaries, not the pieces
- Collaborate effectively
  - inside the boxes, someone else is probably there already
- Focusing on structure not function
  - a set of boundaries provides design context and limits
  - enables qualities
Do you need to do any architecture?

- Do you have a number of independent things cooperating?
- Do you already have a unifying mechanism and structure to combine them?
- No?
- You have “gaps” between the pieces ...

... you could do with some “architecture”
Multi-Application Systems

Order Management System

Inventory Management System

Fulfilment System

Accounts Ledger System

Application integration is all boundaries and gaps!
Where do you focus architecture effort?

- More design problems than you have time to solve
- Many can be (should be) solved by individual teams
- Some need wider context to understand and solve
Where do you focus architecture effort?

Focusing on the boundaries results in a focus on components, connectors, responsibilities and interactions ... architecture!

A good set of boundaries places useful constraints on those working within them
Where does an architect fit with delivery teams?

- A manager?
- Working in one of the teams?
- Across all of the teams?
Where does an architect fit with delivery teams?

Manager(s)

Blue Team

Green Team

Yellow Team

Purple Team

Architecture
Dangers of the Metaphor

• All metaphors have the danger of over use
  • “don’t be stupid”
  • architecture work is not just in the gaps
  • you can’t just worry about the boundaries

• If you only own the boundaries, what do you own?
  • lack of tangible ownership
  • need people to understand and value this work

• Sometimes you need to “get off the fence”
  • into the detail within the boundaries
  • where you’ll find more boundaries you couldn’t see before!
System Qualities and Boundaries
Achieving Qualities

- **Security**
  - threats appear at boundaries so we secure boundaries, connectors and information crossing boundaries
  - security options may require boundaries within a system

- **Performance and Scalability**
  - interactions across boundaries can destroy performance but partitioning is needed for scalability

- **Availability**
  - location of boundaries in software and deployment platform dictate possible availability & resilience

- **Evolution**
  - position and nature of boundaries constrains evolution
Security

• Threats appear at boundaries
• Boundaries and interactions need to be secured
  • choice of boundaries limits the possible security options
Performance and Scalability

- Where do you put the boundaries?

- Leads you to consider trade offs
  - each option affects P&S but also security, evolution, ...
  - each set of boundaries enables or limits deployment options
Availability

• Availability requires replication to allow for failure

• Replication requires layering and modularisation in order to enable it
  • modules (vertical partitions) to allow replication options
  • layers (horizontal partitions) to allow use of replication

• So availability also requires thoughtful placement of boundaries (partitioning)
Each decision about boundaries affects the type and properties of the availability that the system can provide.
Evolution

• Evolution relies on well placed boundaries
  • layers
  • modules
  • encapsulated change

• Too many boundaries make change difficult
  • seemingly innocuous change involves many components

• Lack of solid boundaries makes change impossible
  • the big ball of mud that no one dares to change
Evolution

**Boundaries for the sake of boundaries**

**No thought given to boundaries**

**Boundaries with a Purpose**
Achieving Qualities

• None of these qualities are achieved just by putting the boundaries in the right place

• Focusing on boundaries can guide architecture work
  • most quality properties heavily influenced by concerns at boundaries
  • someone else is working inside the components (hopefully!)
  • keeps the architecture work manageable
Summary
To Conclude

- **Software architecture work can be difficult to define and justify**
  - we know it when we see it

- **Focusing on boundaries is a good metaphor for architecture work**
  - boundaries lead you to fundamental structure, interfaces and interactions – the core of software architecture
  - boundary decisions have fundamental architectural impact

- **Using this metaphor can help to define, justify and focus architecture work**
  - and identify places that architecture work is missing
  - whoever does it
Questions and Comments?

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