Dividing and Conquering at Architectural Scale

Eoin Woods

www.eoinwoods.info

About Me

- Software architect at UBS Investment Bank
  - responsible for equity swaps technology
- Software architect for ~10 years
- Author of "*Software Systems Architecture*" book with Nick Rozanski
  - 2nd edition published in October 2011
- IASA and BCS Fellow, IET member, CEng

We Design at Different Scales

- Organisations - Enterprise Architects
  - organisation wide technical decisions
  - standards, policies, application landscapes
- Systems – Application Architects
  - system wide technical decisions
  - system design, patterns, cross-cutting concerns
- Software Modules – Subsystem Designers
  - localised design decisions within a system
  - design and build of their modules

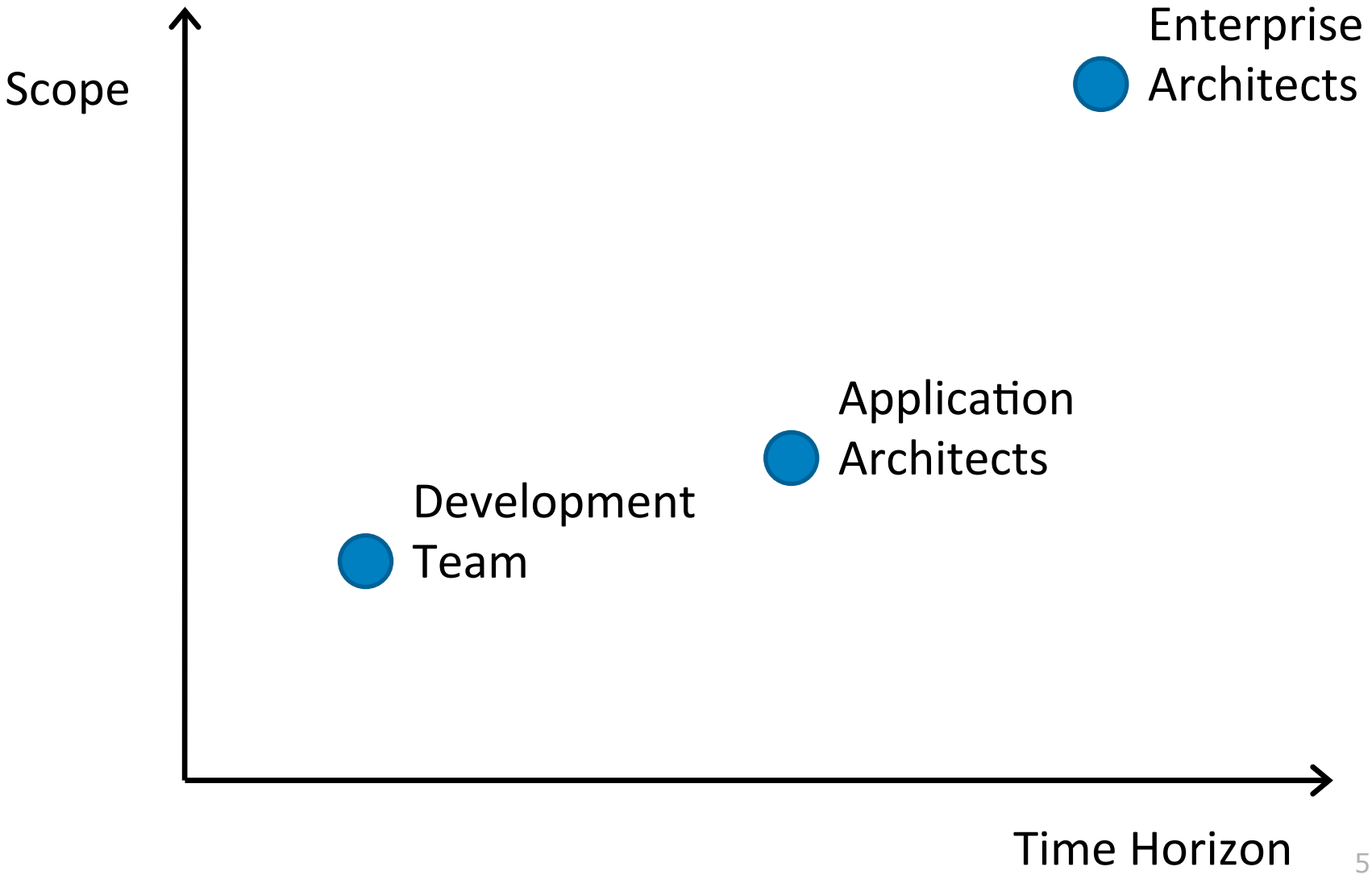
A Common Problem When Scales Collide

EA define strategic *policies and standards* ...

... which application architects find restrictive and so largely ignore as they create *application architectures* ...

... which are often ignored by development teams who are under pressure to get *this release of their system* delivered on time

The Reason - Differing Focus and Priorities



An Example

- EA want systems secured using *standard patterns and security services* with *centrally controlled authorisation*
- Application architects want *standard security*, but want flexible *"mix and match" security services* and want *locally defined and controlled authorisation*
- Teams don't want any of this and want to use simple *local username/password* security with *application defined roles* where needed

Design Principles – A Unifying Concept

- What is a "principle" ?

  a fundamental truth or proposition serving as the foundation for belief or action [OED]

- An architecture principle is ….

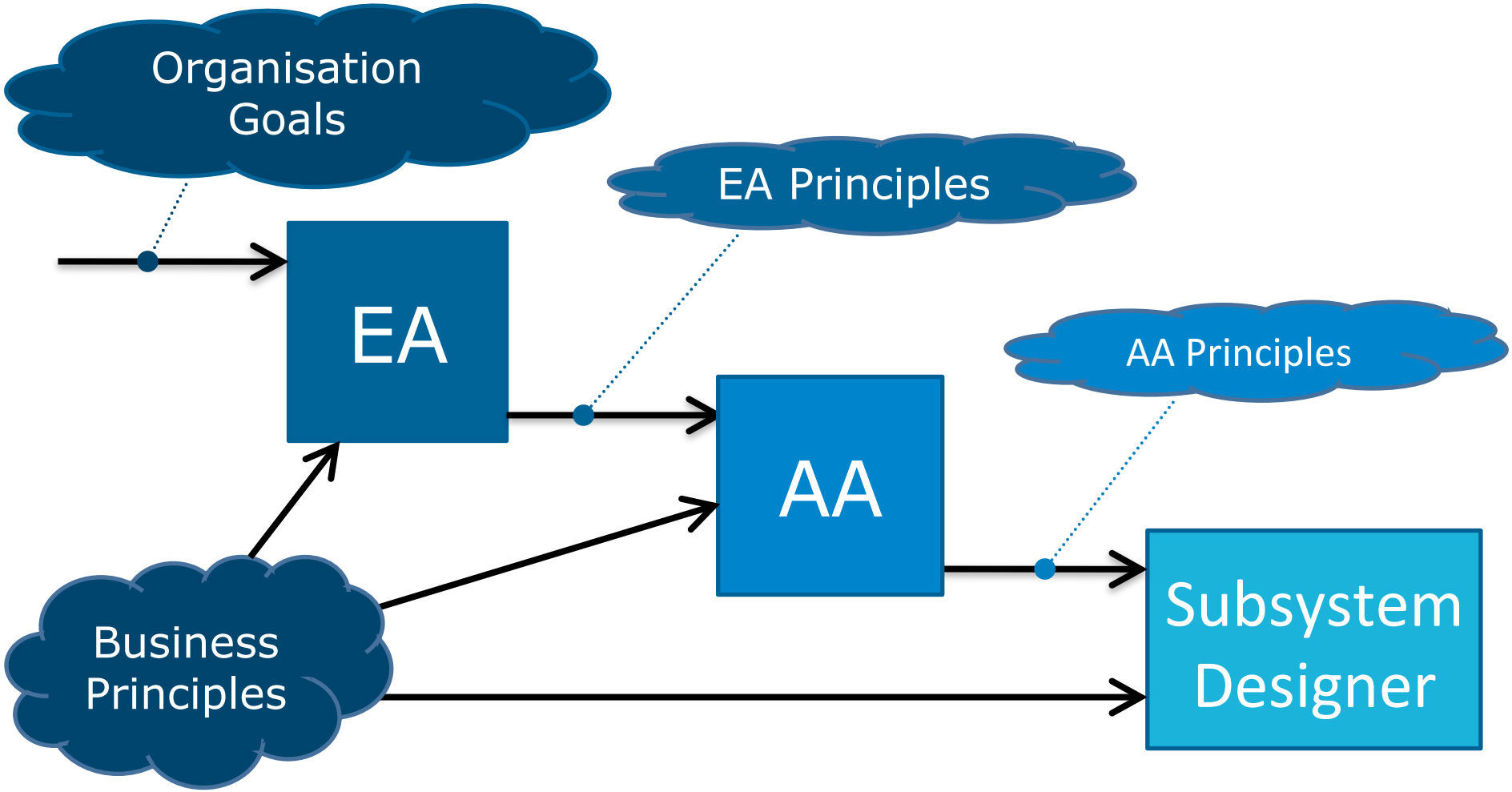
  a declarative statement that normatively prescribes a property of the design of an artefact, which is necessary to ensure that the artefact meets its essential requirements

  [Danny Greefhorst and Erik Proper – 2011]

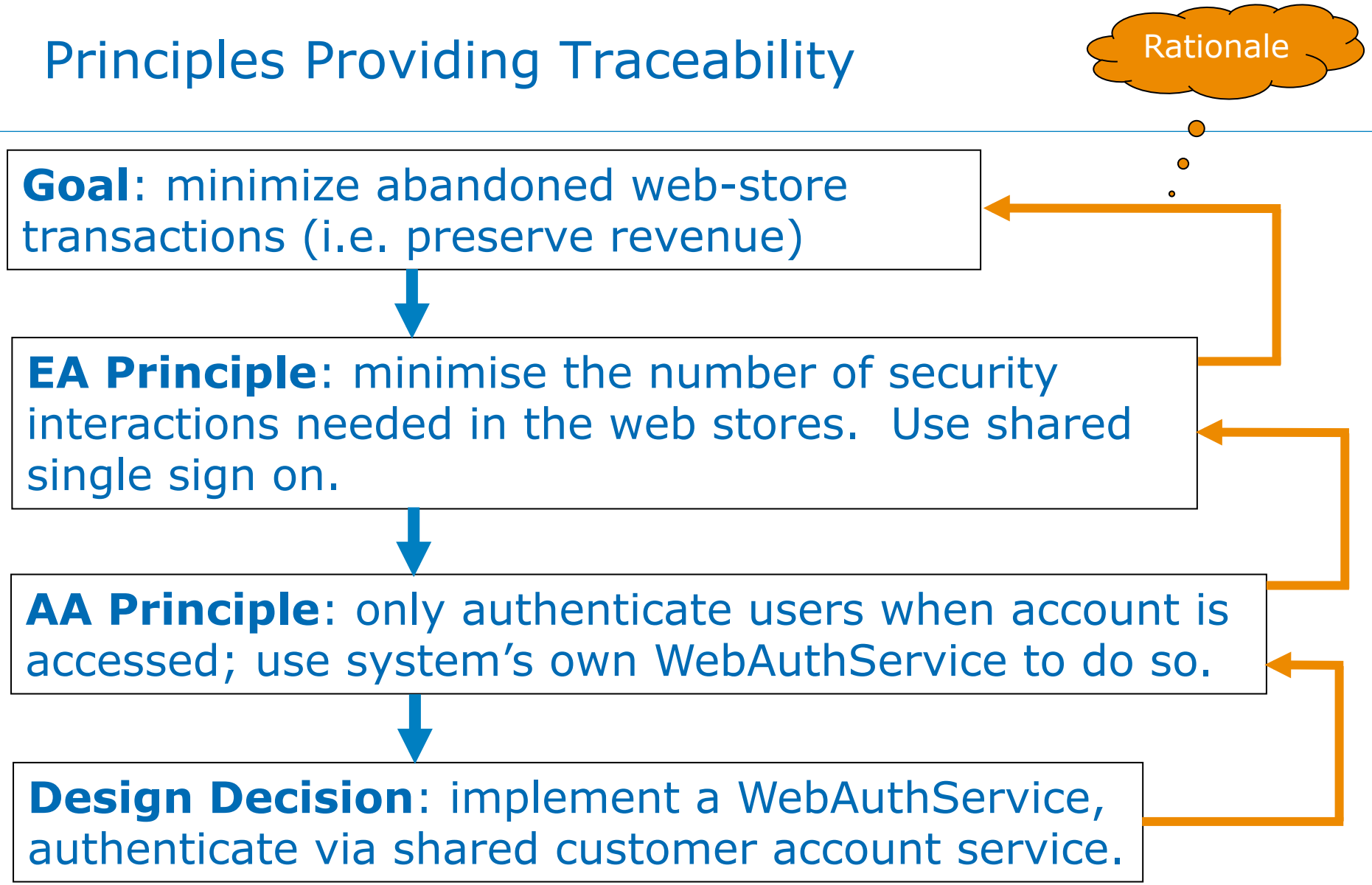
A Tangible Example

- **Use Tiered Architectures**:
  Separate data from applications; applications should employ tiers to separate presentation, logic & data. Our rationale for this is:
  - data lives longer than applications; business logic lives longer than user interfaces
  - tiers separate aspects of the system to allow different rates of evolution
  - tiers allow variation of technology and qualities for different parts of systems (e.g. scalable servers, secure databases)
  - while more complex initially than two tier systems, this is outweighed by the benefits for all but the simplest cases

Principles Unify Across Scales



Principles Providing Traceability



Principles allow a design decision to be traced to a business goal

What do Principles Look Like?

Organisational goal:

G1: build/buy flexibility and long term application vendor flexibility (and are prepared to pay for it)

EA principles:

EP1: no design-time inter-system dependencies

EP2: integrate using a neutral data format

EP3: use 3rd party formats, then organisation's, then system specific

EP4: prefer messaging for integration

[All traceable back to goal G1]

(continued ...)

What do Principles Look Like?

Application architecture principles:

AP1: Use XML messaging (with in-house schema) over pub/sub for external integration [EP2, EP4]

AP2: Define external services using DTO classes not domain classes [EP1]

AP3: Where synchronous integration is essential, use SOAP based web service (using code generator) [EP1 + exception]

The Result of Using Principles

Informed design decisions:

1. Implement `AttributionData` service using local XML schema XML messages over Tibco EMS *[AP1 with exception for local XML schema]*
2. Access `BenchmarkDefinitions` service using "PM-Schema" XML messages over Tibco EMS *[AP1, complies with AP2, AP3]*
3. Retrieve prices via C++ vendor API *[exception required for vendor & system dependency*]

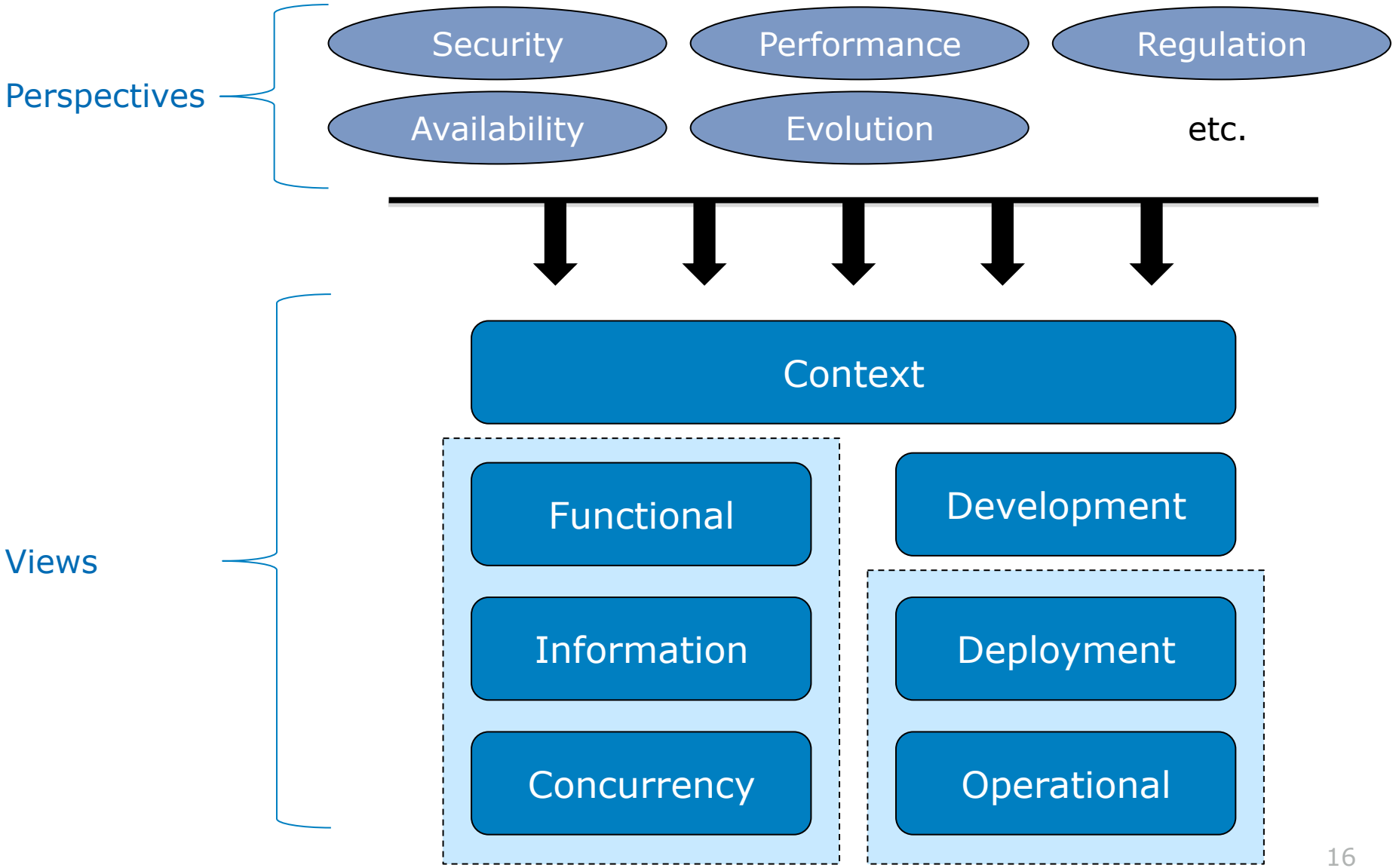
Why Use Design Principles?

- Principles unify the decision making process
  - link decisions made from goals down to software design
- Principles can guide design
  - provide context and constraints for decisions
- Principles can justify decisions, cost and time
  - e.g. principle that all systems must allow for HA deployment
    => need for multi-node software support in designs
- Principles can foster collaboration & shared values
  - if developed in a collaborative fashion

Difficult Aspects of Design Principles

- Identification
  - people find non-trivial principles hard to find (avoid truisms)
- Description
  - difficult to be clear, complete, succinct & understandable
- Validation
  - difficult to validate and measure the value
- Communicating
  - often difficult to understand & internalise

Viewpoints and Perspectives Define Architectures



Principles in Architectural Definition

Principles

Guide Form

Views

Guide Qualities

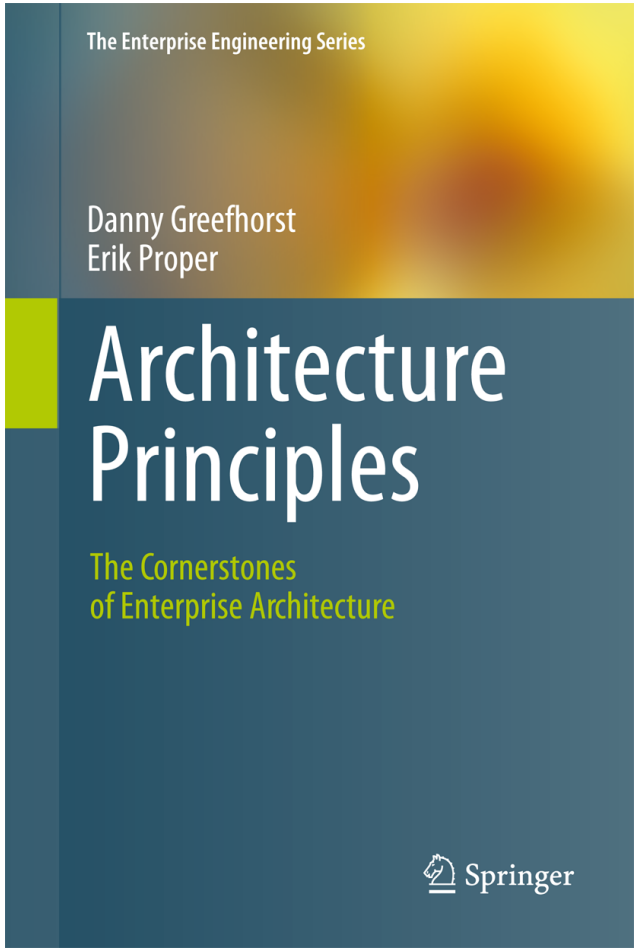
Perspectives

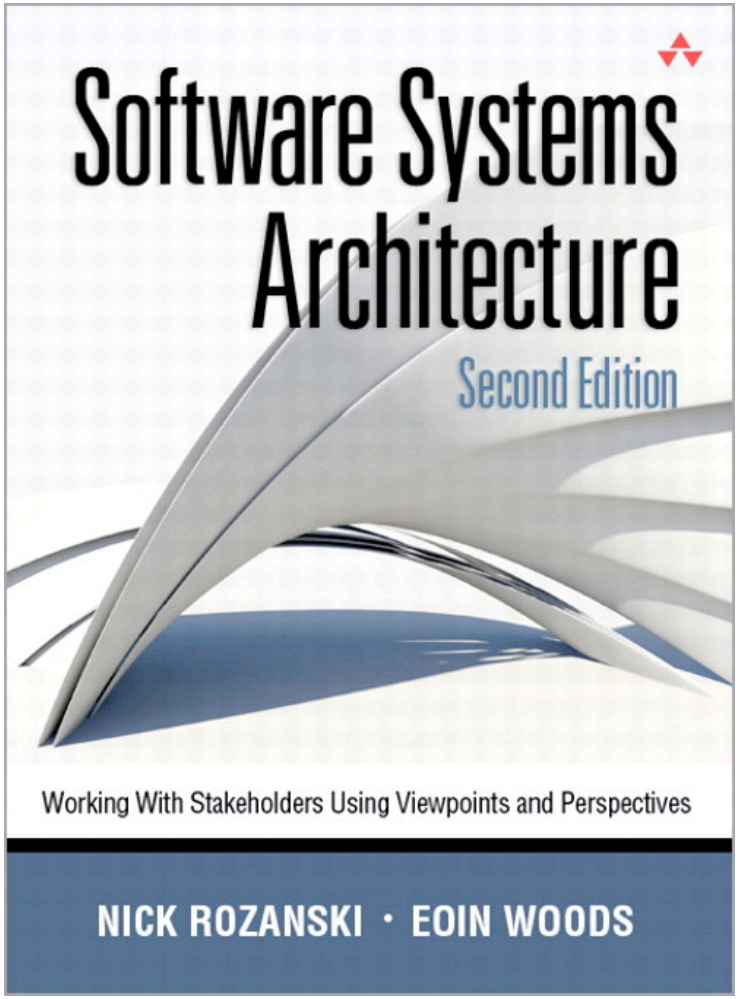
Guide Selection

Summary

- Principles provide “laws” to guide the design process
  - can be used at many different levels of scale
  - help to create informed design decisions
- Principles can provide traceability
  - link back to more abstract principles or underlying goals
  - justifies decisions by reference to a particular context
- Common concept unifies across different scales
  - from business through EA, AA and into software design
  - can guide the use of other architectural techniques

To Learn More





Questions and Comments?

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