Dividing, Conquering and Reuniting with Architecture Principles

Eoin Woods

www.eoinwoods.info

About Me

- Software architect at UBS Investment Bank
  - responsible for equity swaps technology
- Software architect for ~15 years
  - participated in EA work for the last 5 years
- Author of "*Software Systems Architecture*" book with Nick Rozanski
  - 2nd edition published in October 2011
- IASA and BCS Fellow, IET member, CEng, CITP

Our Problem – Designing at Different Scales

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

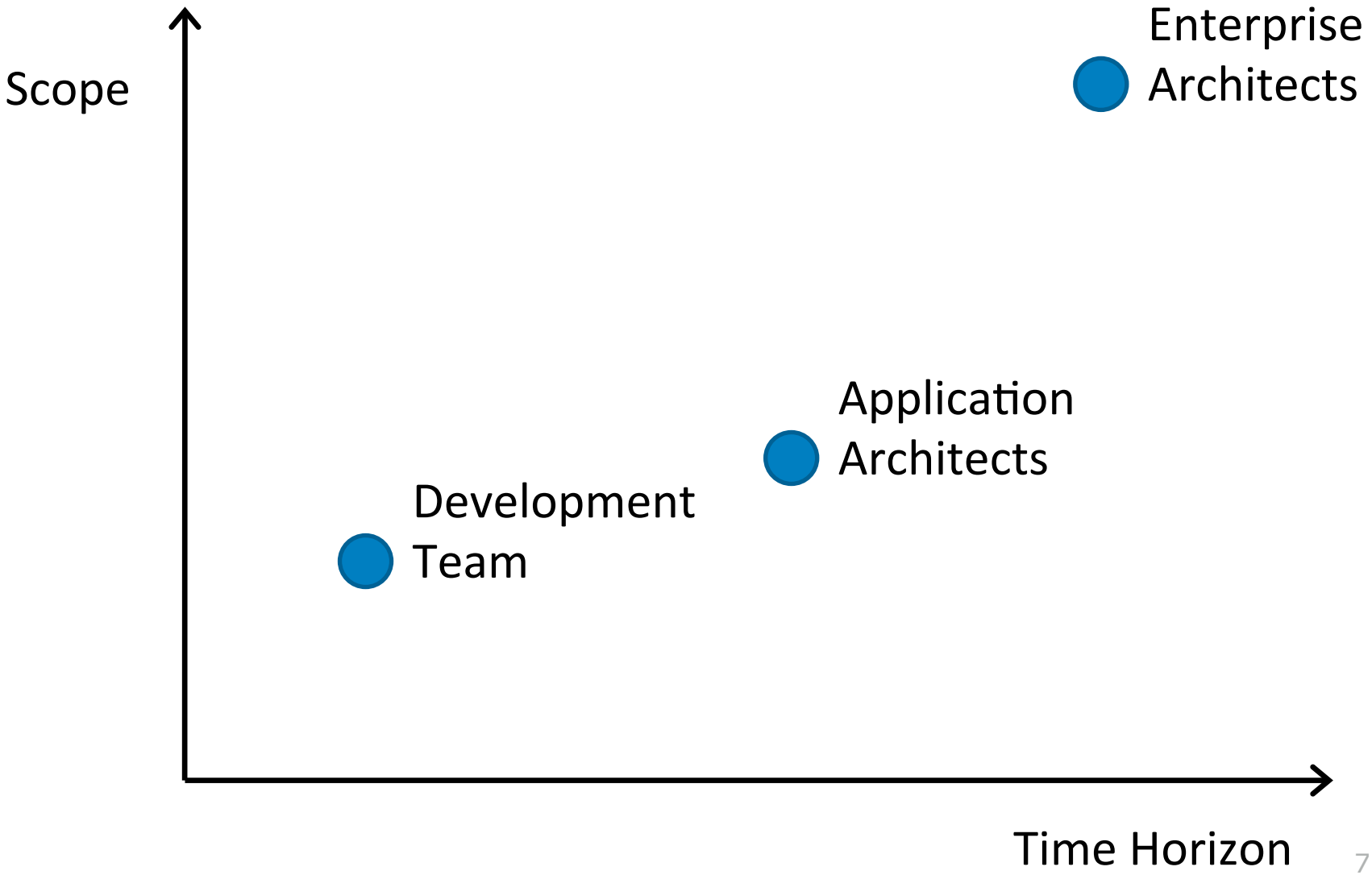
A Common Problem When Scales Collide

EA define strategic *policies, standards, roadmaps* ...

... 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



The Reason - Differing Scope and Priorities

EA	• long term cost/quality/general time to market • organisation wide scope • aligning with & supporting organisation goals
AA	• long term cost/quality/system delivery time • single system scope • intra-system standardisation
Teams	• short term cost/quality/system delivery time • single system scope • standardisation only for development speed

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

Architecture Principles

Architecture 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 ….

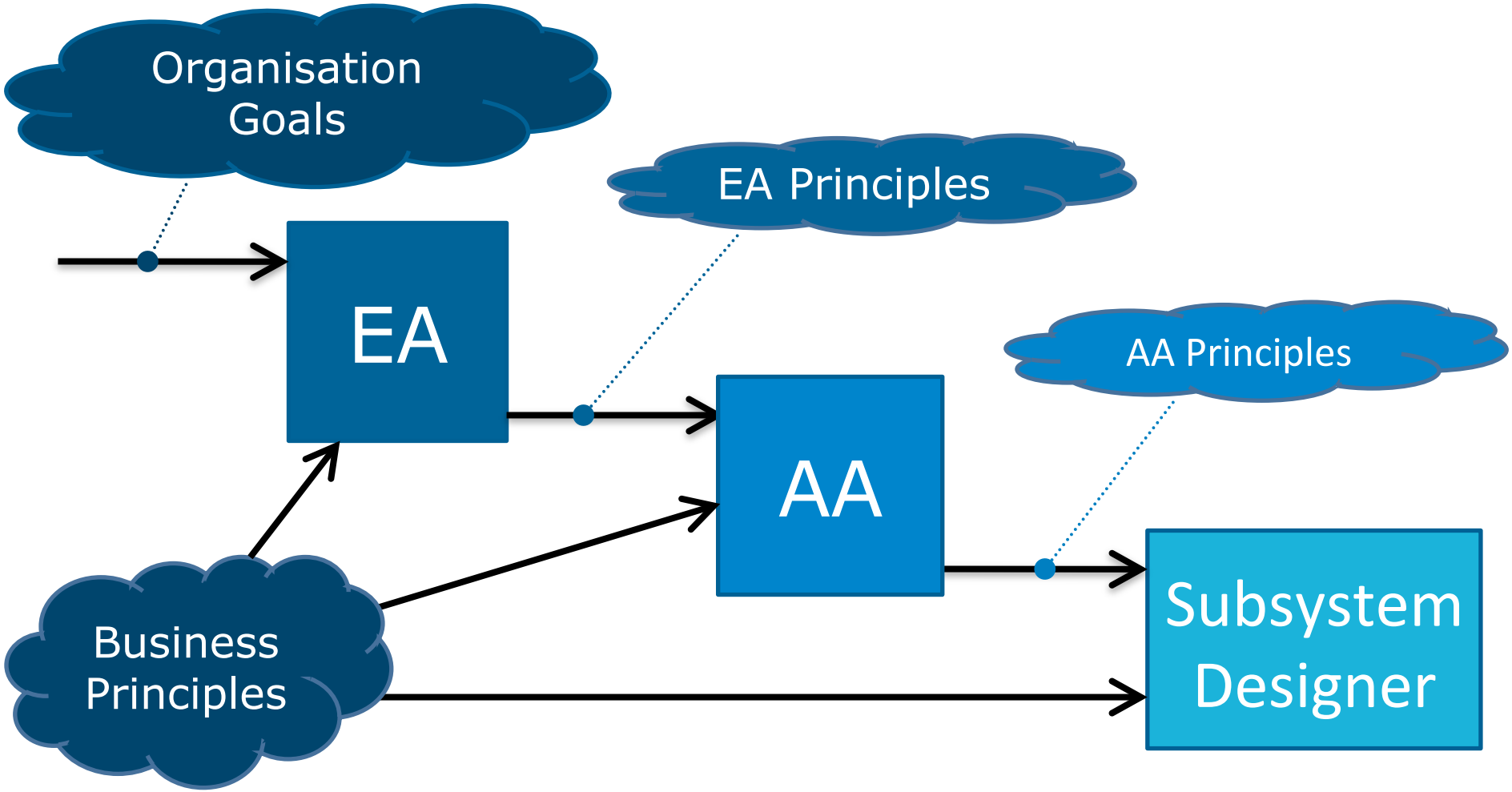
  a declarative statement that normatively prescribes a property of the design of an artefact, which is necessary to ensure that the artefact meets its essential requirements

  [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

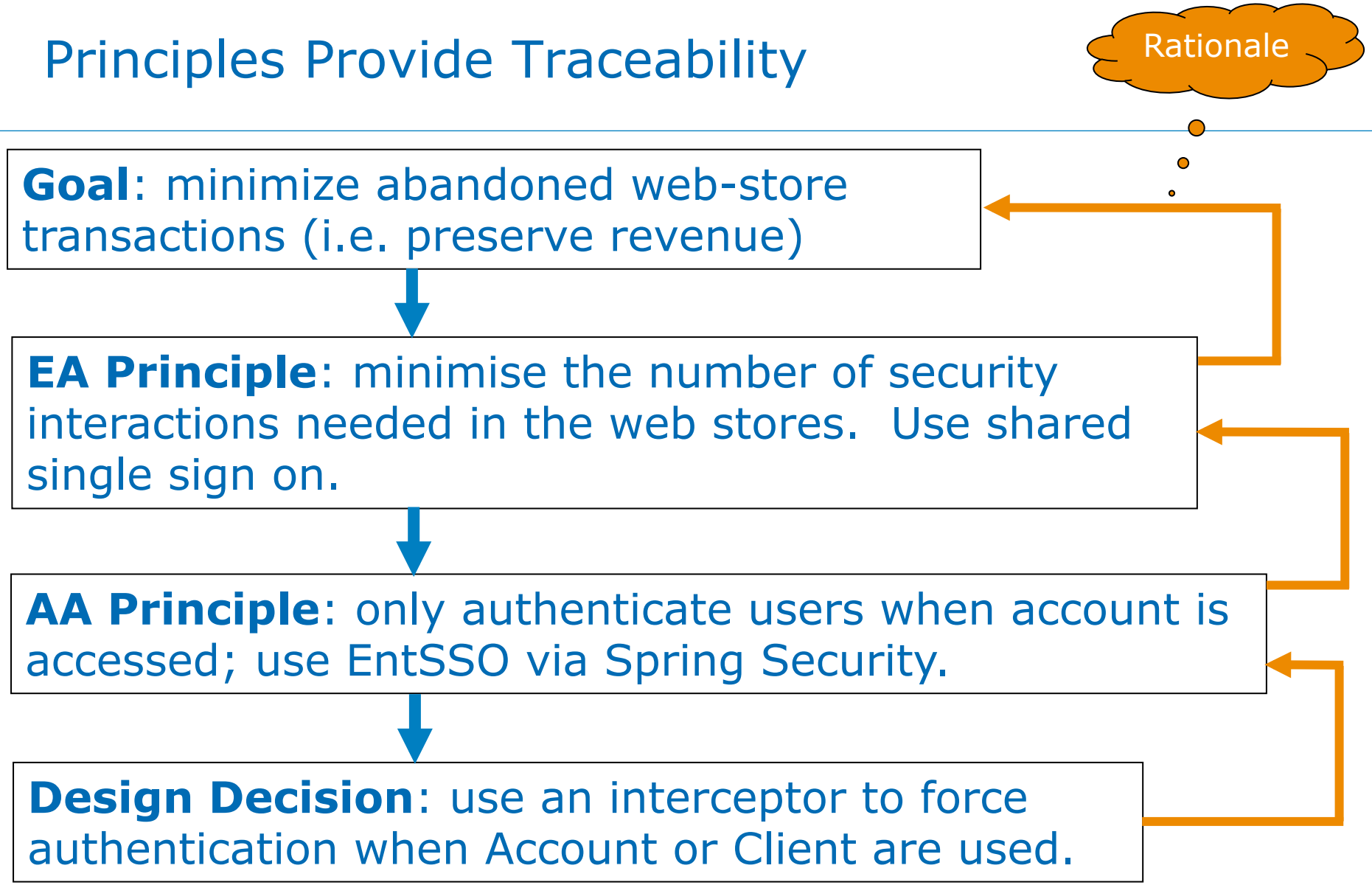
Principles Unify Across Scales



Principles Unify Across Scales

EA	**use** business and organisational principles and priorities to **create** EA principles and design decisions
AA	**use** EA principles and business principles to **create** application architecture principles and design decisions
Teams	**use** application architecture principles and business principles to **create** design decisions

Principles Provide 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: inter-system build-time dependencies must be limited to an adapter layer

EP2: integrate using a neutral data format

EP3: prefer 3rd party formats, then organisational, then system specific

EP4: prefer messaging for integration

[All traceable back to goal G1]

What do Principles Look Like?

Application architecture principles:

AP1: Use XML messaging (with in-house schema) over pub/sub for external integration [EP2, 3 & 4]

AP2: Define external services using DTO classes and adapters, don't reveal domain classes [EP1]

AP3: Where synchronous integration is essential, use SOAP based web service and an adapter to the local domain model
[exception case – supports EP1]

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*]

Architecture Principles in Practice

Types of Architecture Principles

- Define a goal
  - "single customer logon for all of our web sites"
- Indicate a preference
  - "prefer 3rd party data formats, over in-house, over custom"
- Avoid a specific technical problem
  - "use tiers to avoid UIs becoming bound to databases"
- Encourage a way of working
  - "don't repeat yourself" (DRY) [H&T]
- Remind people of useful proven observations
  - "abstractions live longer than details" [H&T]

Properties of Good Principles

Constructive	stated for a definite purpose, useful guide for decision making
Reasoned	rational, logical, consistent
Well Articulated	comprehensible by all of the necessary stakeholders
Testable	possible to check if you've followed it and where the exceptions are
Significant	not just a truism; would the opposite *ever* be the case?

[Nick Rozanski]

Why Use Architecture 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

When to Violate a Principle

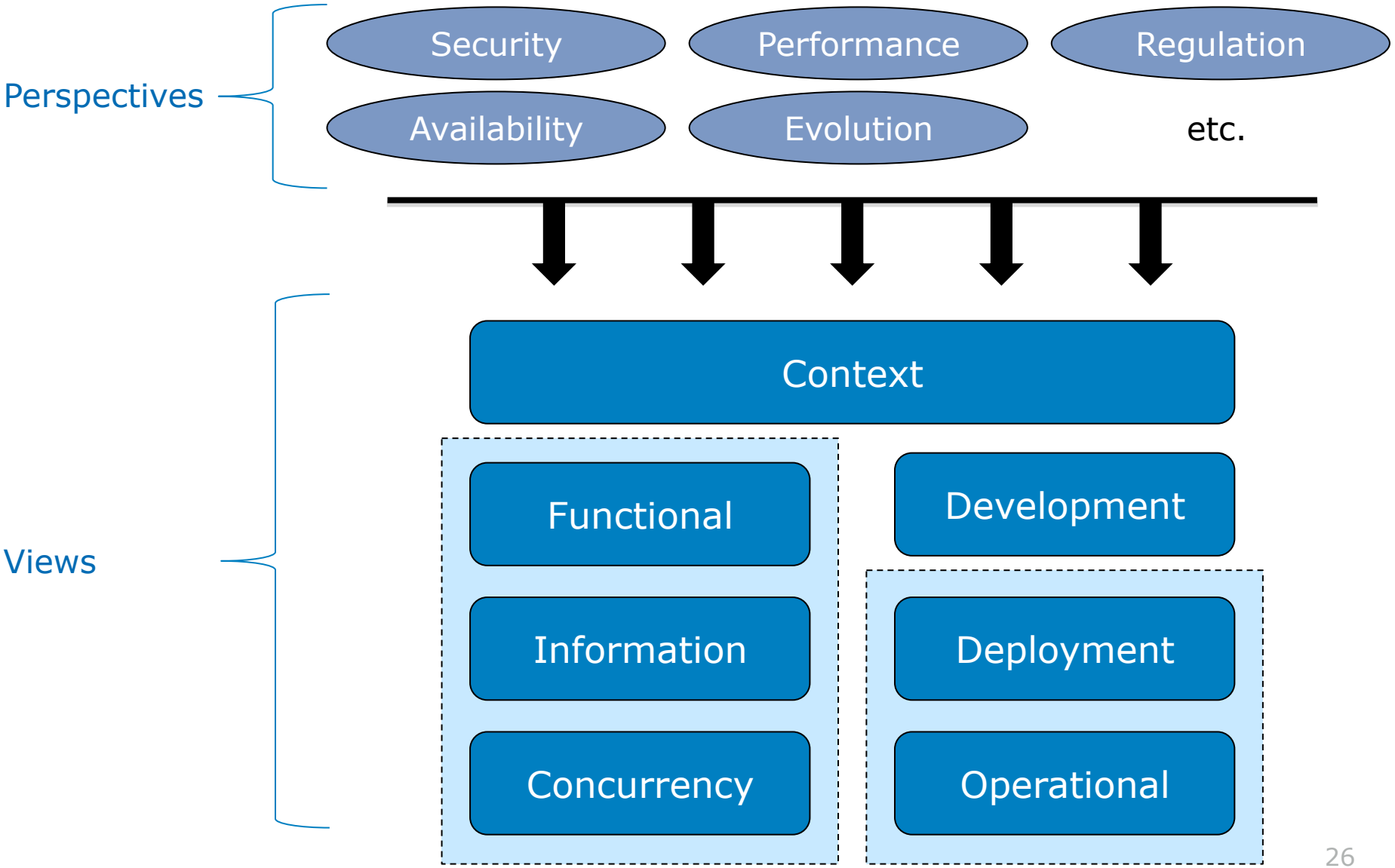
- Principles can’t always be followed
  - but when broken must be broken for justifiable reasons
  - i.e. benefits have to outweigh the costs
- This doesn’t (necessarily) reduce their usefulness
  - reason for breaking a principle is valuable design information
  - a large number of violations signal the need to revisit the principle concerned
  - capturing the violation signals the non-standard nature of the decision

Difficult Aspects of Architecture 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
- Retrofitting
  - can be very difficult to apply principles after the fact

Architecture Principles in Context

Viewpoints and Perspectives Define Architectures



Principles in Architectural Definition

Principles

Guide Form

Views

Guide Qualities

Perspectives

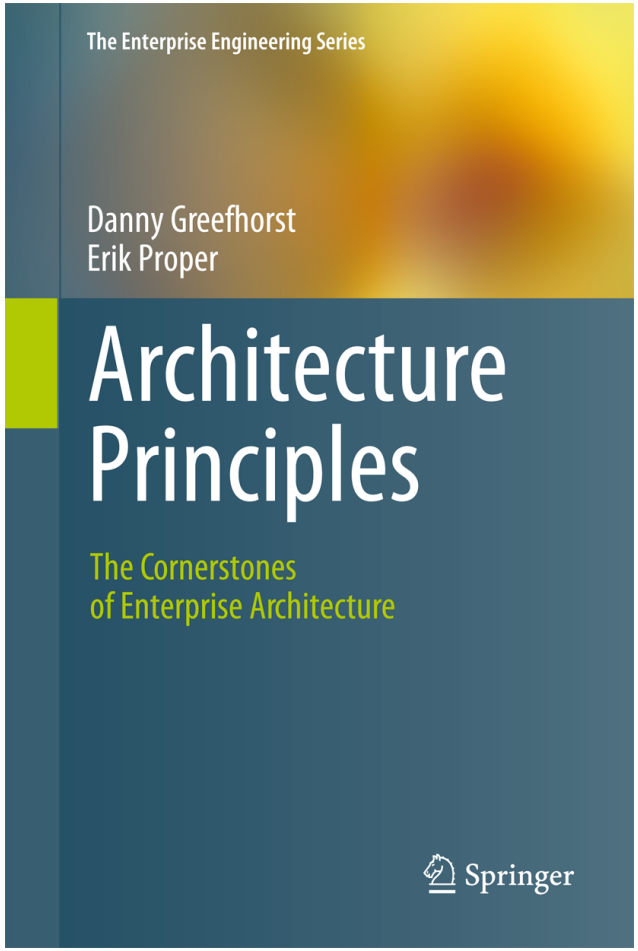
Guide Selection

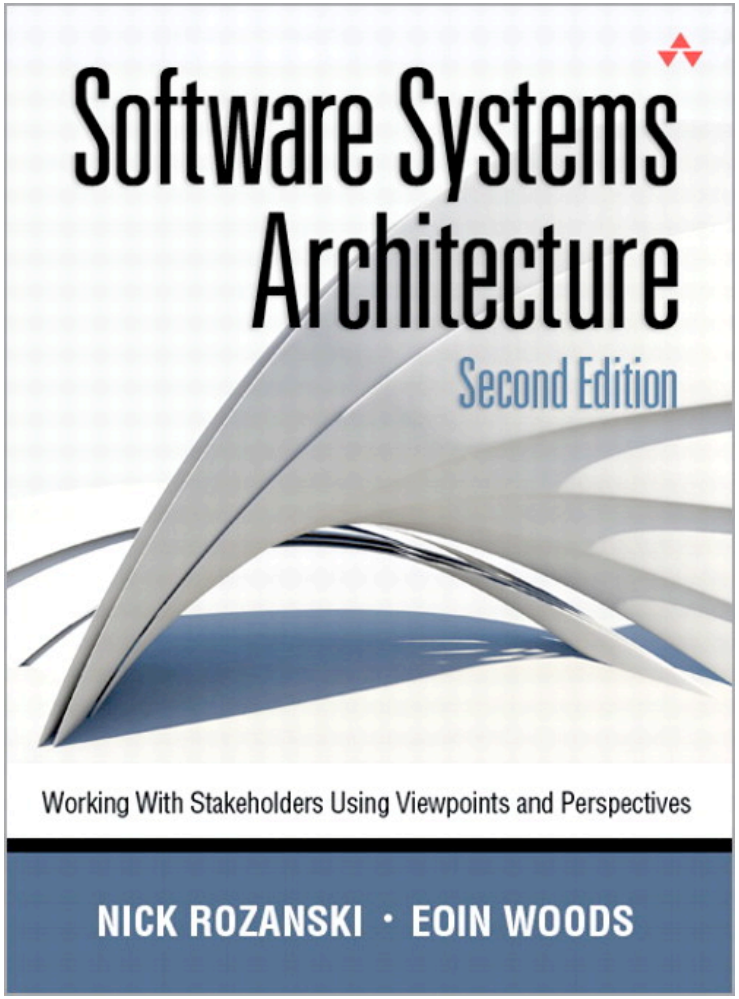
Summary

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