Using Design Principles to Unify Architecture and Design

Eoin Woods
Barclays Global Investors

www.eoinwoods.info
www.barclaysglobal.com/careers

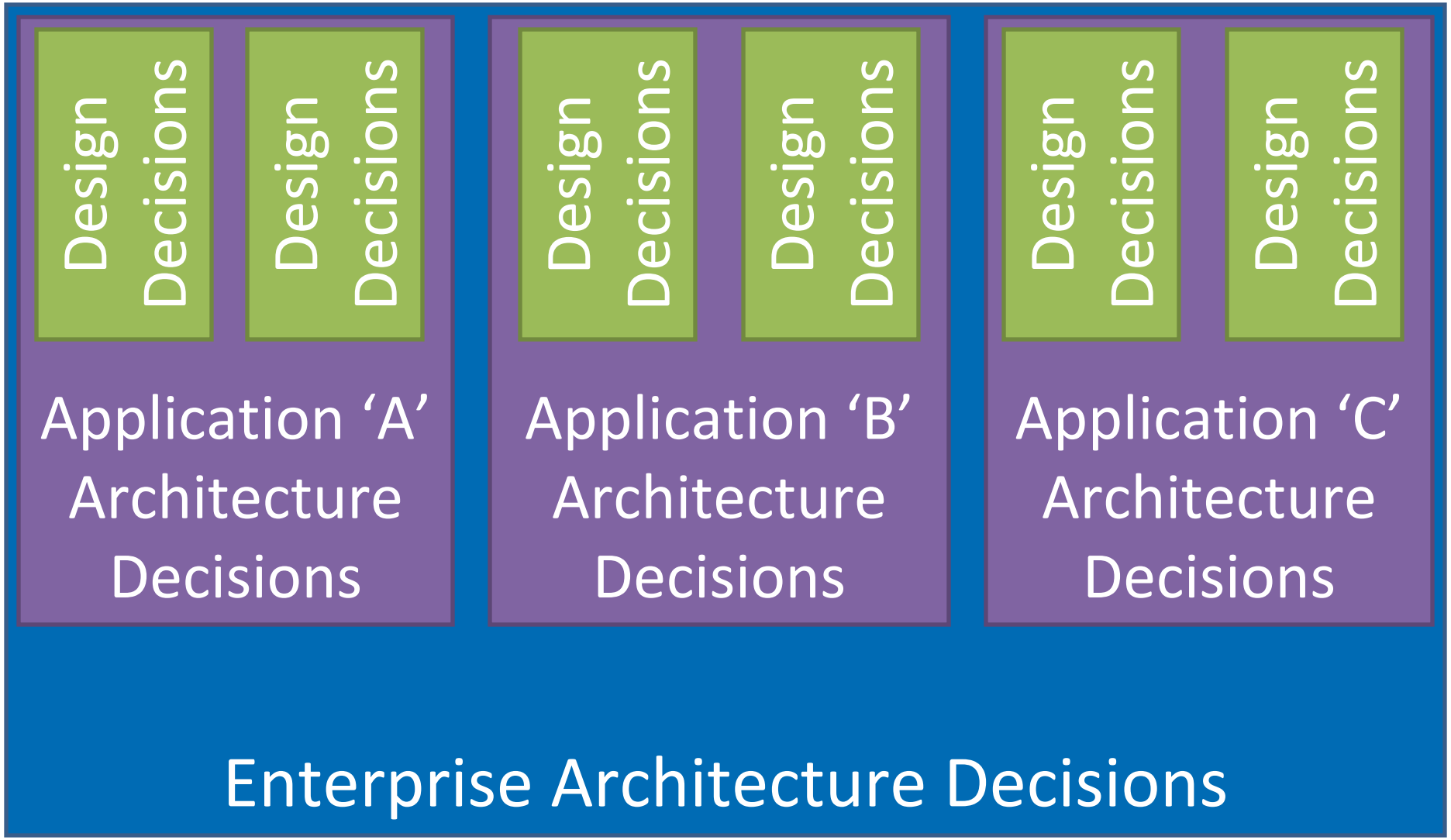
About Me

- Software architect at Barclays Global Investors
  - head of the Application Architecture group
  - aligned with Equities and Capital Markets area
  - responsible for Apex, a new portfolio management system
- Software architect for ~10 years
- Author of "*Software Systems Architecture*" book with Nick Rozanski
- IASA and BCS Fellow, IET member, CEng

Software Development Tribes

- Enterprise Architects
  - organisation wide technical decisions
  - standards, policies, application landscapes
- Application Architects
  - system wide technical decisions
  - system design, patterns, cross-cutting concerns
- Development Teams
  - all local design decisions with a system
  - oh, and all the real work!

EA, AA and Development Teams



A Common Problem

EA define strategic *policies and standards* …

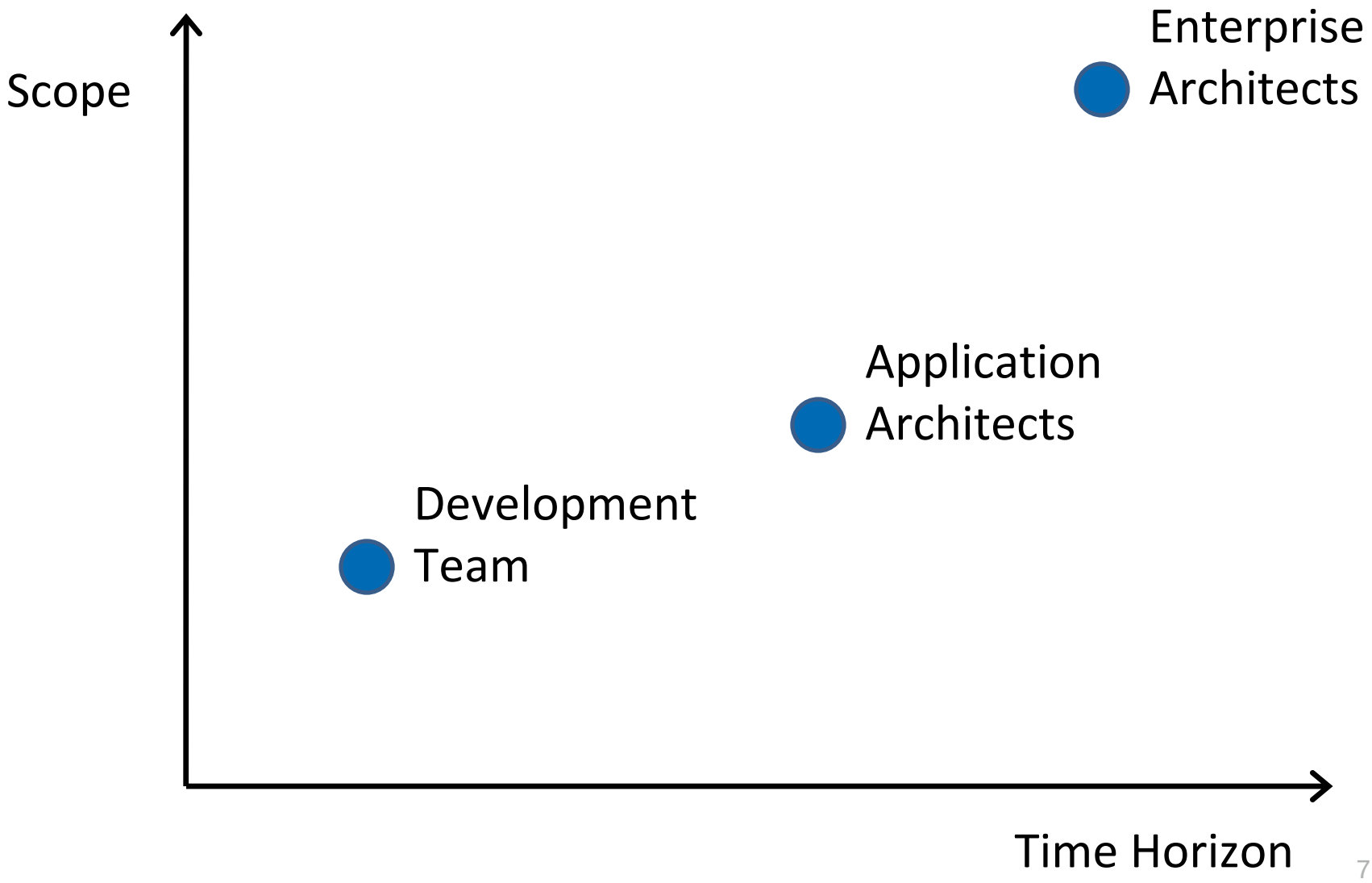
… which application architects find restrictive and so largely ignore as they create *application architectures* …

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

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

The Reason - Differing Focus and Priorities



An Example

- EA want systems linked via *standard patterns and middleware* with a *service catalogue*
- Application architects want *easy integration*, but *don't want a service catalogue* and want to select and *vary details by system*
- Teams *don't want any of this* and want to get data into their systems *as easily as possible* (e.g. remote database access)

Underlying Problem

- Differing priorities are caused by a *lack of common understanding*
- *AA doesn't understand* what is guiding *EA decision making*
- *Developers don't understand* what is guiding *AA decision making* (let alone EA decisions!)
- No concept being used to communicate context & rationale
- Decision making separated from implementation

What could we do to fix this?

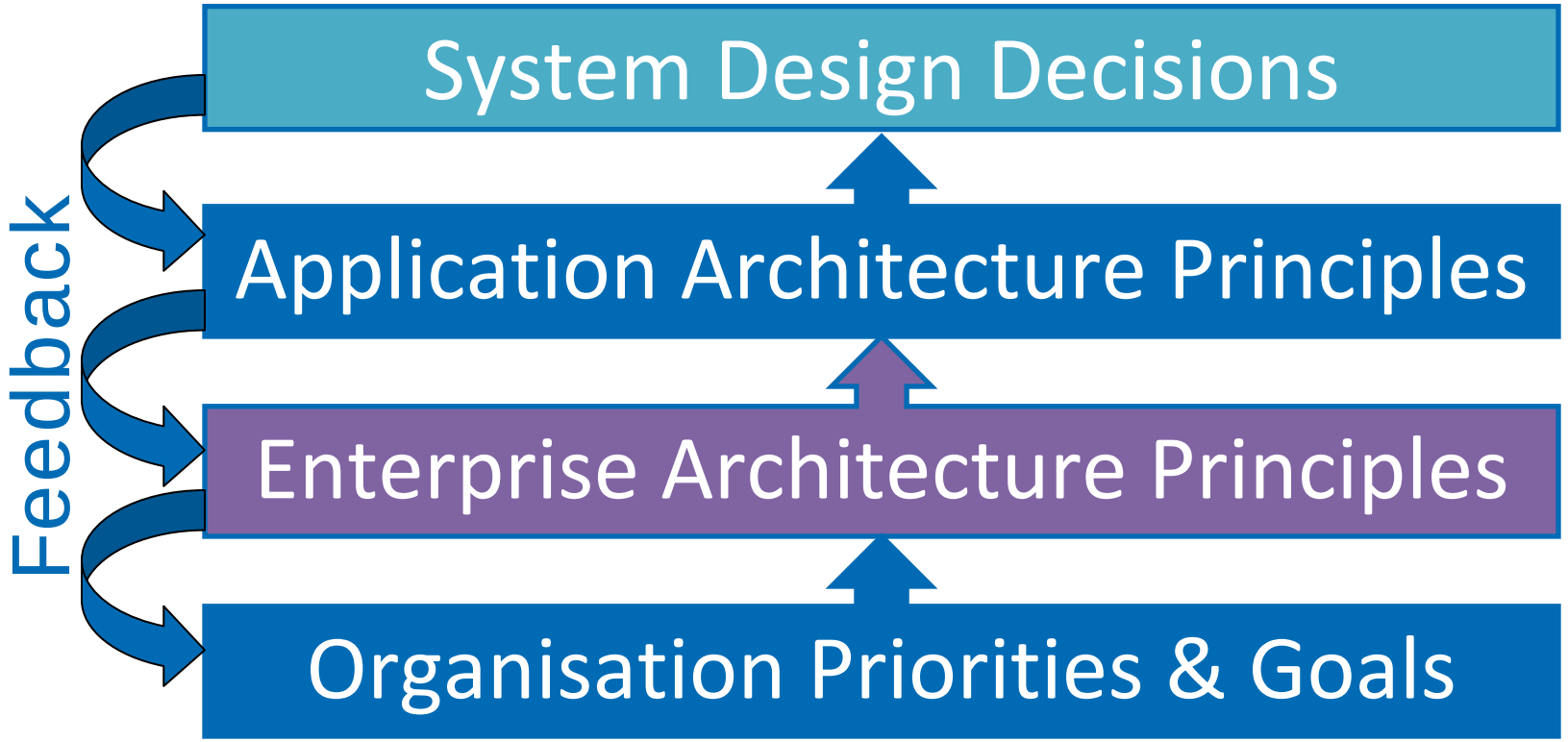
Design Principles

- What is a “principle” ?
  - a fundamental truth or proposition serving as the foundation for belief or action [OED]
  - a comprehensive and fundamental law, doctrine or assumption [Webster's]
- So a design principle is a fundamental “truth” or “law” that serves as the foundation for design action (i.e. guides design decisions)
  - a unifying concept for software development?

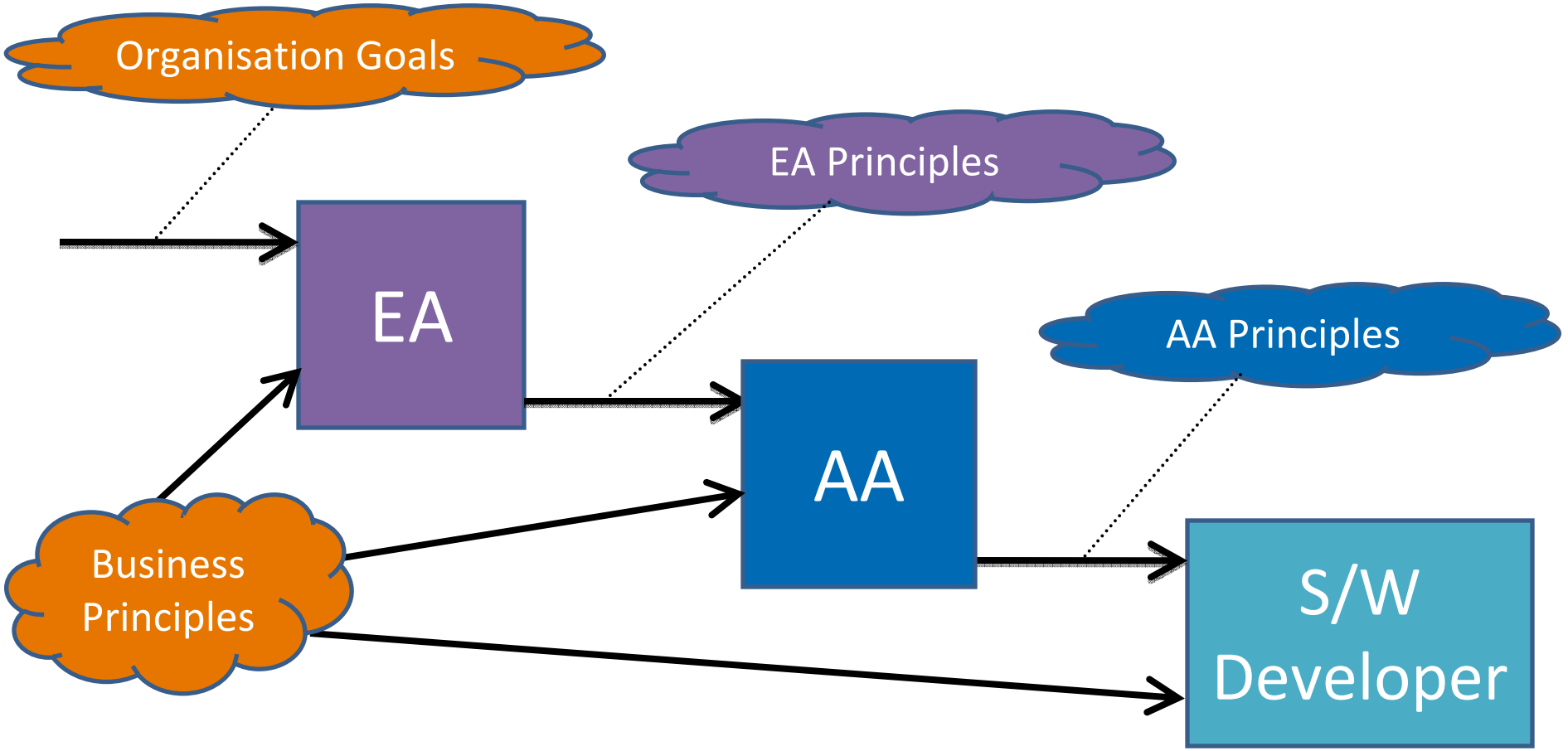
Aside: Principles vs. Patterns vs. Decisions

- Decision
  - makes a concrete design decision
  - is bound to a specific design context
- Pattern
  - makes a set of concrete design decisions
  - is unbound, but with applicability defined
- Principle
  - places a constraint on design decisions
  - is unbound, but may need applicability defined

Design Principles in Context



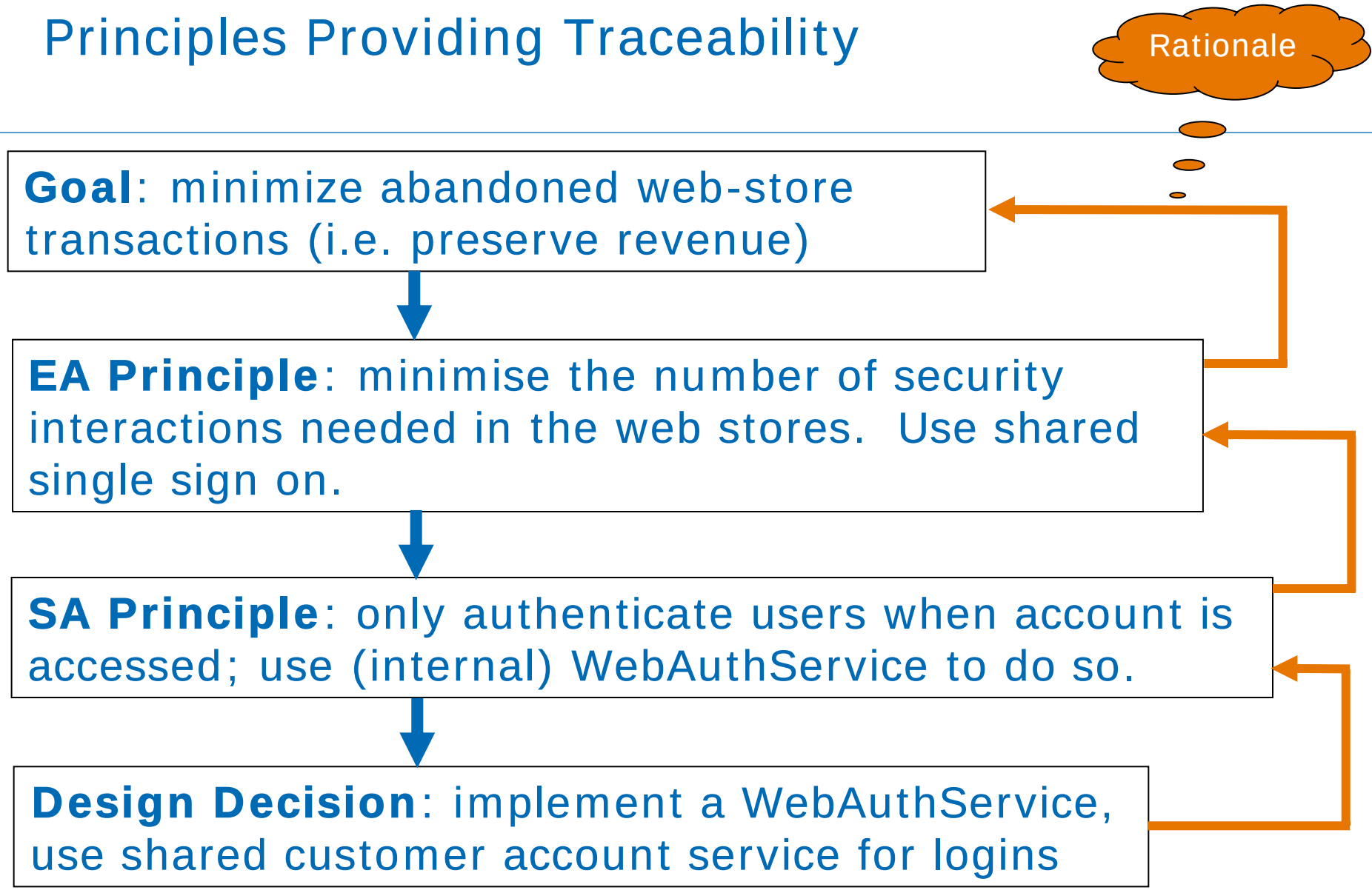
Principles as a Unifying Concept



Principles as a Unifying Concept

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 Providing Traceability



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

What do Principles Look Like?

- Organisational goal:
  - **G1**: we want to have build/buy flexibility and long term application vendor flexibility (and are prepared to pay for it)
- EA principles:
  - **EP1**: avoid design-time inter-system dependencies
  - **EP2**: integrate using a neutral data format
  - **EP3**: use 3rd party formats, then ours, then system specific
  - **EP4**: prefer messaging over RPC for integration

  [All traceable back to goal G1]

(continued ...)

What do Principles Look Like?

- Application architecture principles:
  - **AP1**: Use in-house schema XML messaging 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:
  - Implement `AttributionData` service using local XML schema XML messages over Tibco EMS
    [AP1 with exception for local XML schema]
  - Access `BenchmarkDefinitions` service using PM-Schema XML messages over Tibco EMS
    [AP1, AP2, AP3]
  - Retrieve prices via C++ vendor API
    [exception required for vendor & system dependency]

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

Types of Design 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
  - “identify what varies then encapsulate it” [GoF]
- 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]

Good Design 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 Design Principles?

Why not just capture design decisions or patterns?

Pattern or Decision	Principle
Concrete decision	A constraint on decisions
Fully defined	Minimally constraining
Define an action	Aid understanding
Specific to context	General as possible
Solves a single problem	Guides future decisions

decisions and patterns give people solutions;
principles help them design their own

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
  - e.g. need for multi-node software support from principle that all systems must allow for HA deployments in the future
- Principles can justify costs and time
  - this will take longer, but we understand the underlying goal
- Principles should be developed collaboratively
  - so achieving buy-in, neutrality & good coverage

Difficult Aspects of Design Principles

- Identification
  - people find non-trivial principles hard to find (avoid truisms)
  - examples and experience needed
- Description
  - difficult to be clear, complete, succinct & understandable
- Validation
  - very difficult to know if you have the right set
  - difficult to know if they'll be valuable
- Communicating
  - often difficult for people to understand & internalise
  - finding the right customer

Fruitful Research Topics

- Identification
  - where do principles come from?
  - why do people find them hard to articulate?
- Representation
  - how do you write a principle down?
  - how do you put it in a database and use it?
- Validation
  - what makes a good principle?
  - are principles really valuable? why? how valuable?

Teaching Implications

- What are the implications for the education and training of software engineers?
  - understanding of principles
  - identification of principles
  - representation of principles
  - use of principles in architecture and design
  - are there standard sets that can be taught?

Summary

- Principles provide “laws” to guide the design process
  - can be used at many different levels
  - less constraining than patterns or decisions
- Principles should provide traceability
  - links back to more abstract principle or an underlying goal
  - justifies decisions by reference to a particular context
- Common concept allows unification through design
  - from business through EA, AA and into software design
- A lot to do in order to make principle use widespread
  - work needed in capture, analysis, representation & education

Questions and Comments?

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