Using Design Principles to Unify Architecture and Design

SA2010
London
October 2010

Eoin Woods

www.eoinwoods.info

About Me

- Software architect at BlackRock
  - head of the Application Architecture group
  - also responsible for the architecture of one of the portfolio management systems
- Software architect for ~10 years
- Author of "*Software Systems Architecture*" book with Nick Rozanski

Content

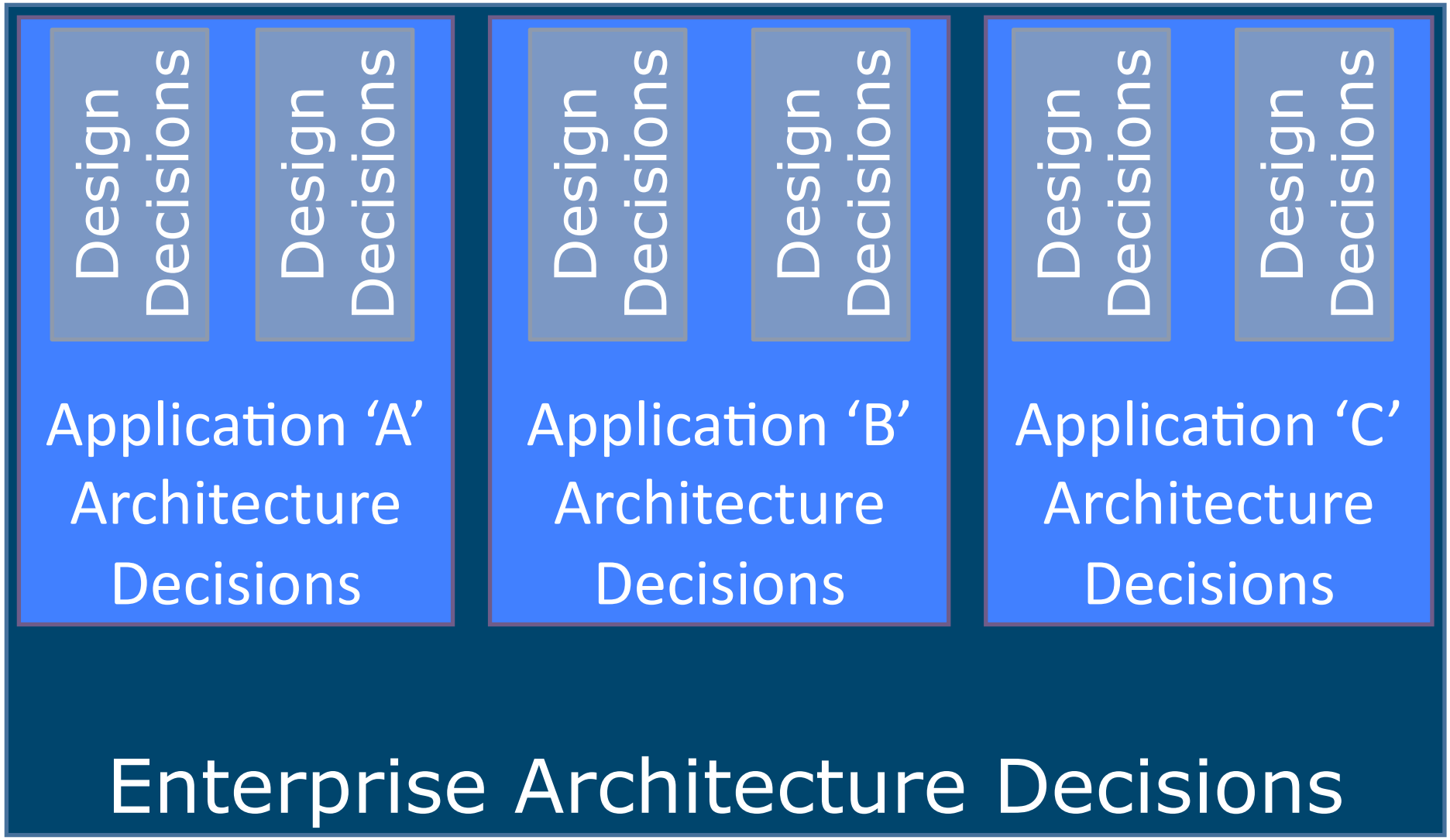
- The Enterprise Software Development Landscape
- Introducing Design Principles
- Using Design Principles
- Design Principles in Practice
- In Conclusion

The Enterprise Software Development Landscape

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* ...

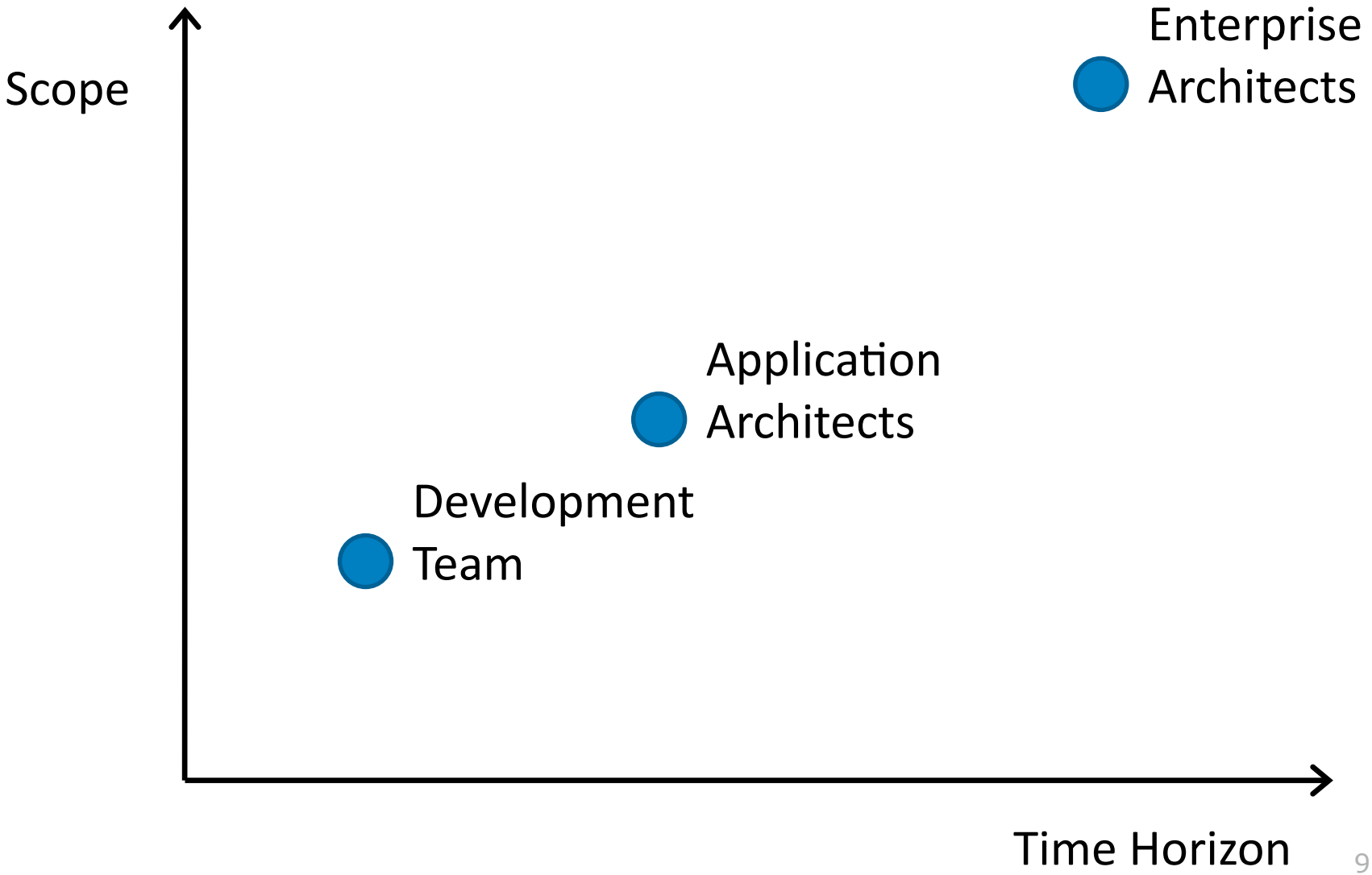
... which application architects find restrictive and so largely ignore as they create *application architectures* ...

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

Cause - 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

Cause - 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?

Introducing Design Principles

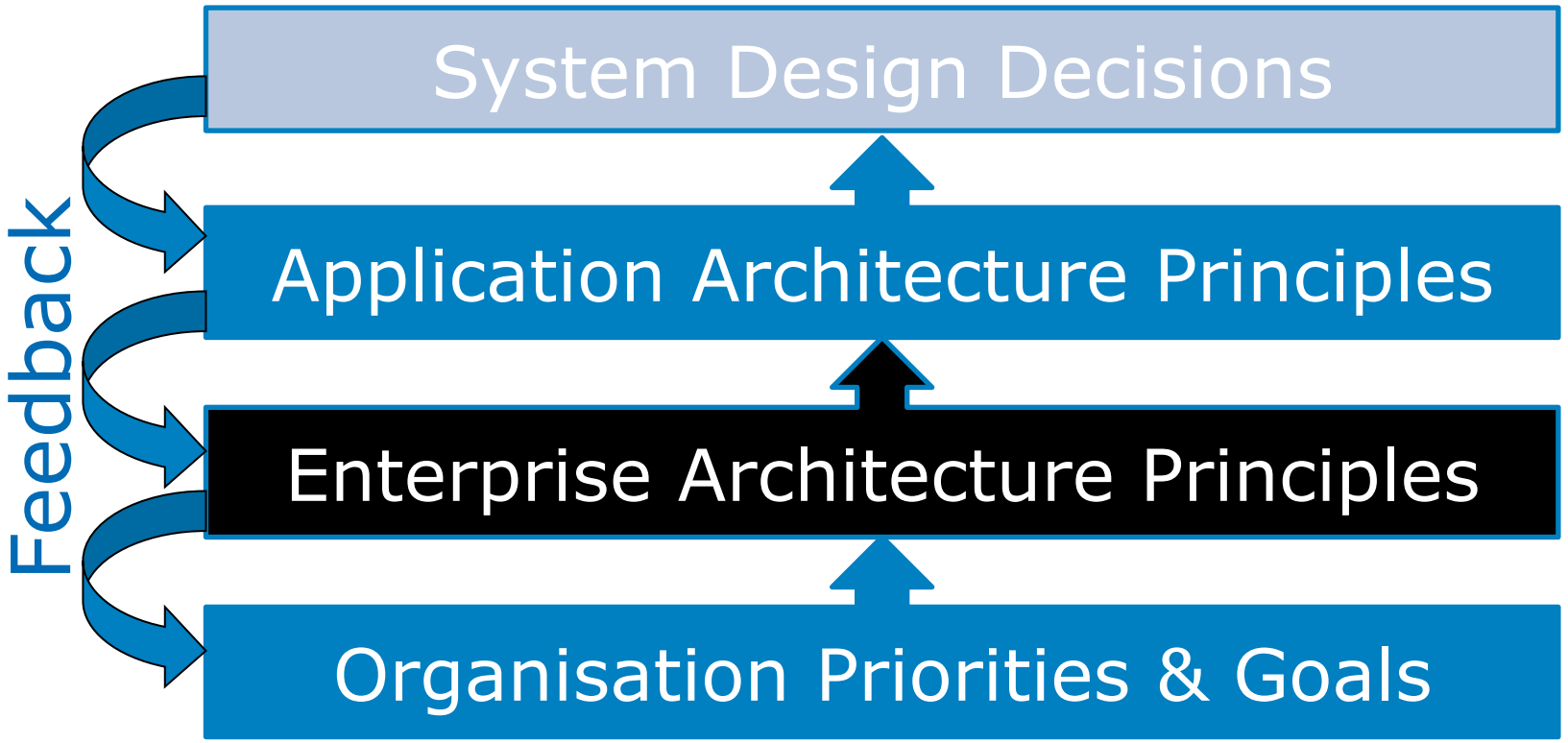
Design Principles

- 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?

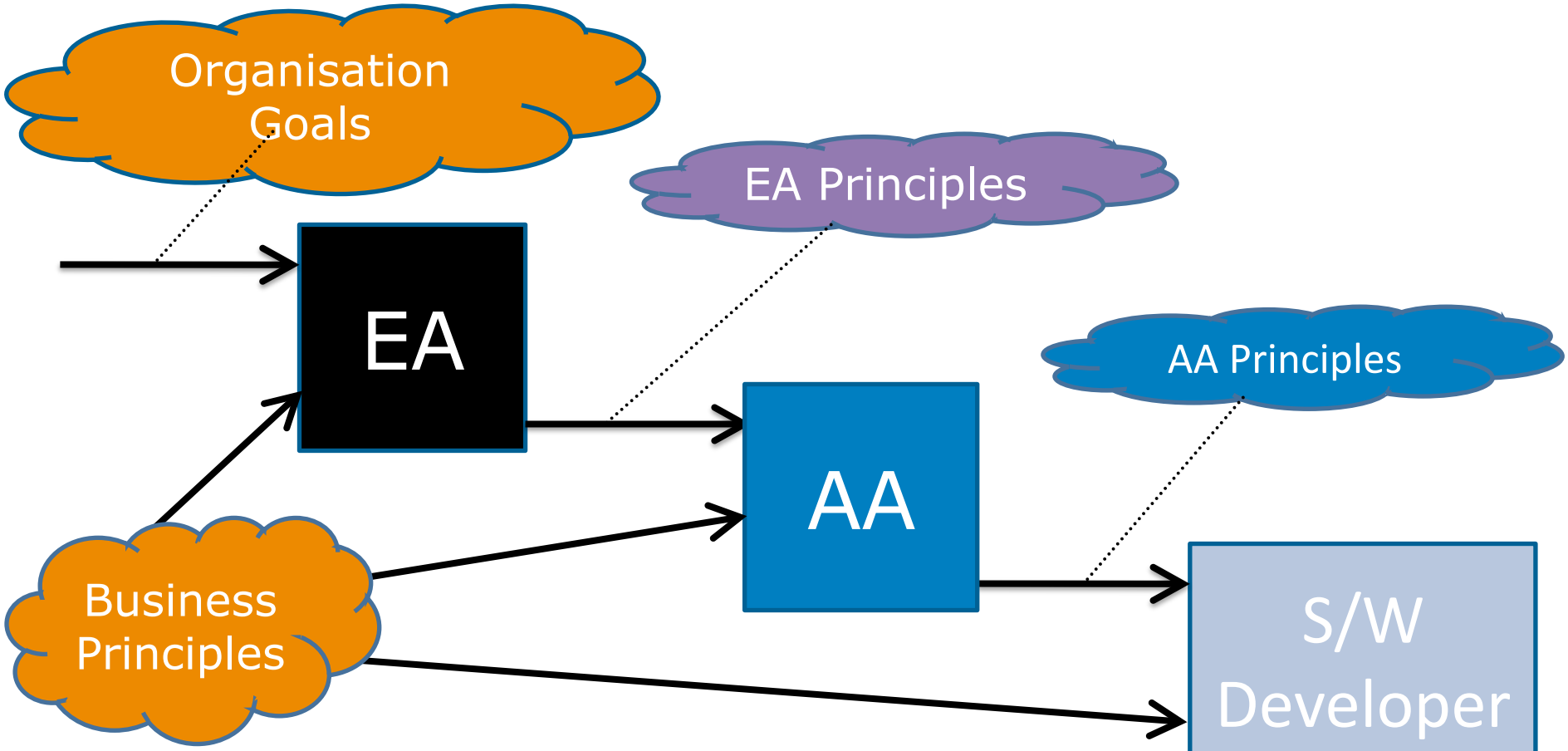
Aside: Principles, Patterns & Decisions

- 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



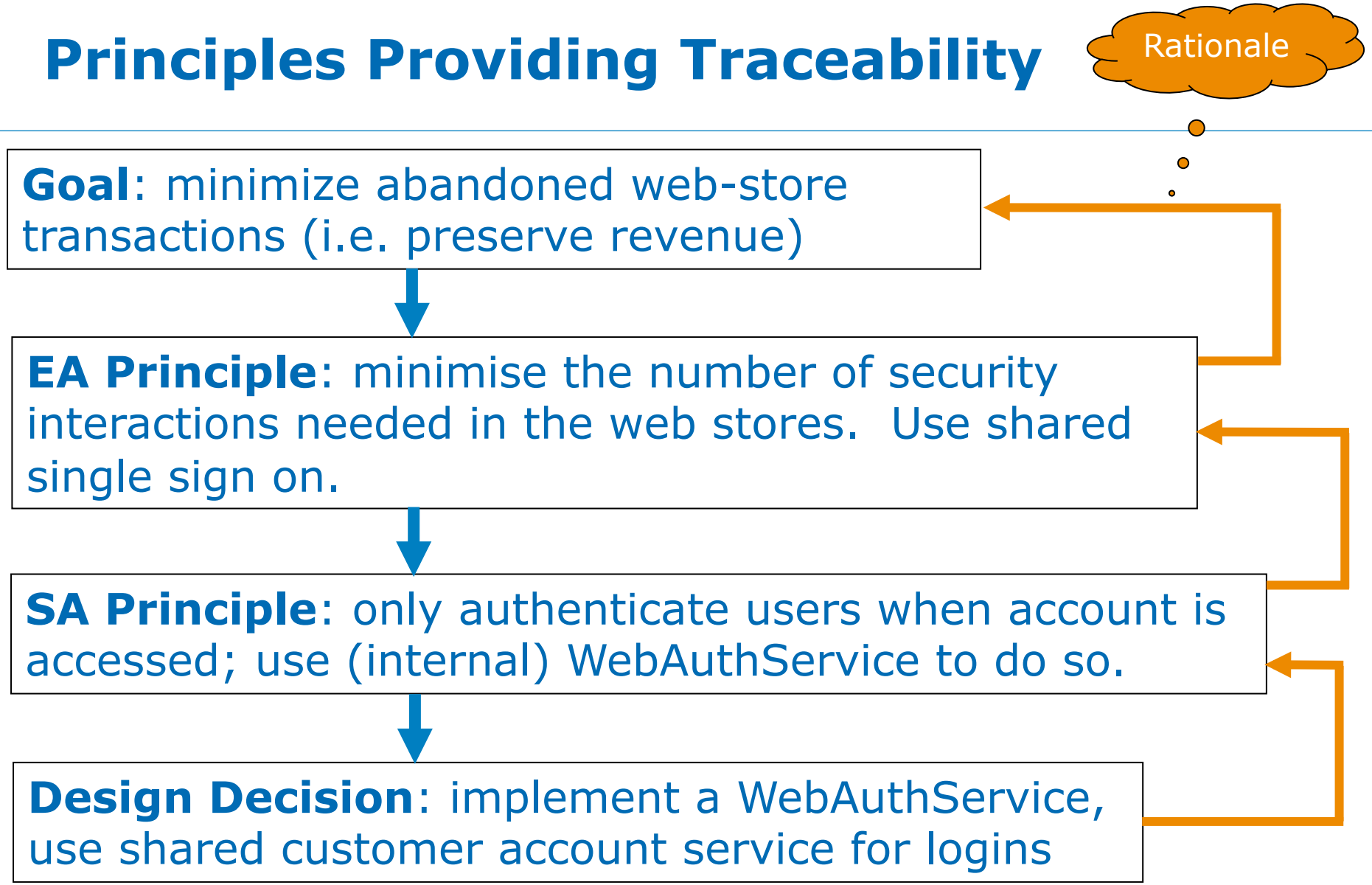
Principles as a Unifying Concept



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

Using Design Principles

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
Prescriptive	specific, providing definite guidance
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

Design Principles in Practice

Using Design Principles in Practice

- Design principles can bring a lot of benefits
- So why doesn't every project use them?
- A couple of reasons ...
  - It's quite difficult to create good sets of principles
  - Identifying coherent sets of principles takes time
  - The pay off is often some way away
  - The concept is understood, but not what it means in practice
- Let's look these obstacles & how to overcome them

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

Identifying Good Design Principles

- Start by reviewing existing design principles
- Collect inputs from your organisation
  - mission statements, plans, visions, roadmaps
  - backlog priorities
  - lists of top problems that key people perceive
- Identify & validate a small set of principles
  - avoid truisms (check: could the negation ever make sense?)
  - keep them specific, precise and prescriptive
  - focus on the rationale, tie back to your organisation
- Validate them by *using* them
  - Do they help you make decisions?

Examples of Good Design Principles (1)

- "Use Layered Architectures"
  - Data must be separated from applications. Applications must conform to an n-tier architecture, separating data from application and business logic wherever possible. This allows the portability of data and supports the sharing of data.

    The initial design maybe more complex, although for anything but a single user system this is soon outweighed by the benefits.
- "Maximise Genericity"
  - always pick the most general solution to a problem that is suitable for the situation being considered, while avoiding premature generic solutions (unstable, unused).

Examples of Good Design Principles (2)

- "Value Simplicity"
  - Applications should be as simple as possible and based on mainstream technologies. Important factors when choosing technologies are their likely longevity, resulting maintainability and access to skills. This impacts the Total Cost of Ownership
- "Keep Dependencies Acyclic" [Bob Martin]
  - no cycles in the dependency graph of software elements
  - avoids design, deploy and runtime problems
- "Dependencies must reflect stability" [Bob Martin]
  - dependencies between elements in a design should be in the direction of the stability of the elements
  - elements should only depend upon elements more stable than they are

Examples of Poor Design Principles

- "Balance performance and security"
  - poorly expressed (too vague, not prescriptive enough)
  - not really testable
  - something of a truism
- "Implement domain services as EJBs in WebLogic"
  - this is just a decision or constraint
  - doesn't provide context, rationale or guide other decisions
- "User interface must support the end user's tasks"
  - the classic truism!

Sources of Design Principles

- Bob Martin's set
  - `www.objectmentor.com/resources/publishedArticles.html`
- Hunt and Thomas
  - *The Pragmatic Programmer: Journeyman to Master* book
- Gamma et al
  - *Design Patterns* book
  - `www.artima.com/lejava/articles/designprinciples.html`
- IBM Design Principles Checklist
  - `www-01.ibm.com/software/ucd/designconcepts/designbasics.html`
- Piero Campanelli
  - very nice list consolidating many sources
  - `tinyurl.com/37grmbb`

Sources of Design Principles

- Industrial Design
  - Universal Principles of Design – Lidwell, Holden & Butler
    - e.g. Consistency - “The usability of a system is improved when similar parts are expressed in similar ways”
- Your organisation
  - mission statements, strategies, roadmaps
  - enterprise architecture principles

In Conclusion

Getting Started

- Stop and work out what is important to your project
  - organisational information
  - design priorities
  - generic principles that particularly apply
- Try to write down 3 principles
  - critique them, share them, revise them
  - use the criteria we've suggested to focus them
  - start with existing principles and tailor them
- Use these principles when making decisions
  - where they don't apply you'll suddenly see others that do
- Slowly build a set specific to your projects

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