Avancier Methods (AM)

Adaptive Architecture

A first draft or manifesto v8

There is an address for comments at the end

It is illegal to copy, share or show this document
(or other document published at http://avancier.co.uk)
without the written permission of the copyright holder

Copyright Avancier Limited 2011

Contents

- The challenge
- **Adaptive architecture techniques**
- 20 Questions about adaptive architecture
- Relaxing version control

Copyright Avancier Limited 2011

Adaptive architecture techniques

Avancier

Modularity is (of course) the central idea.

If you want systems to be adaptive and enable agile change, then you'll need to:

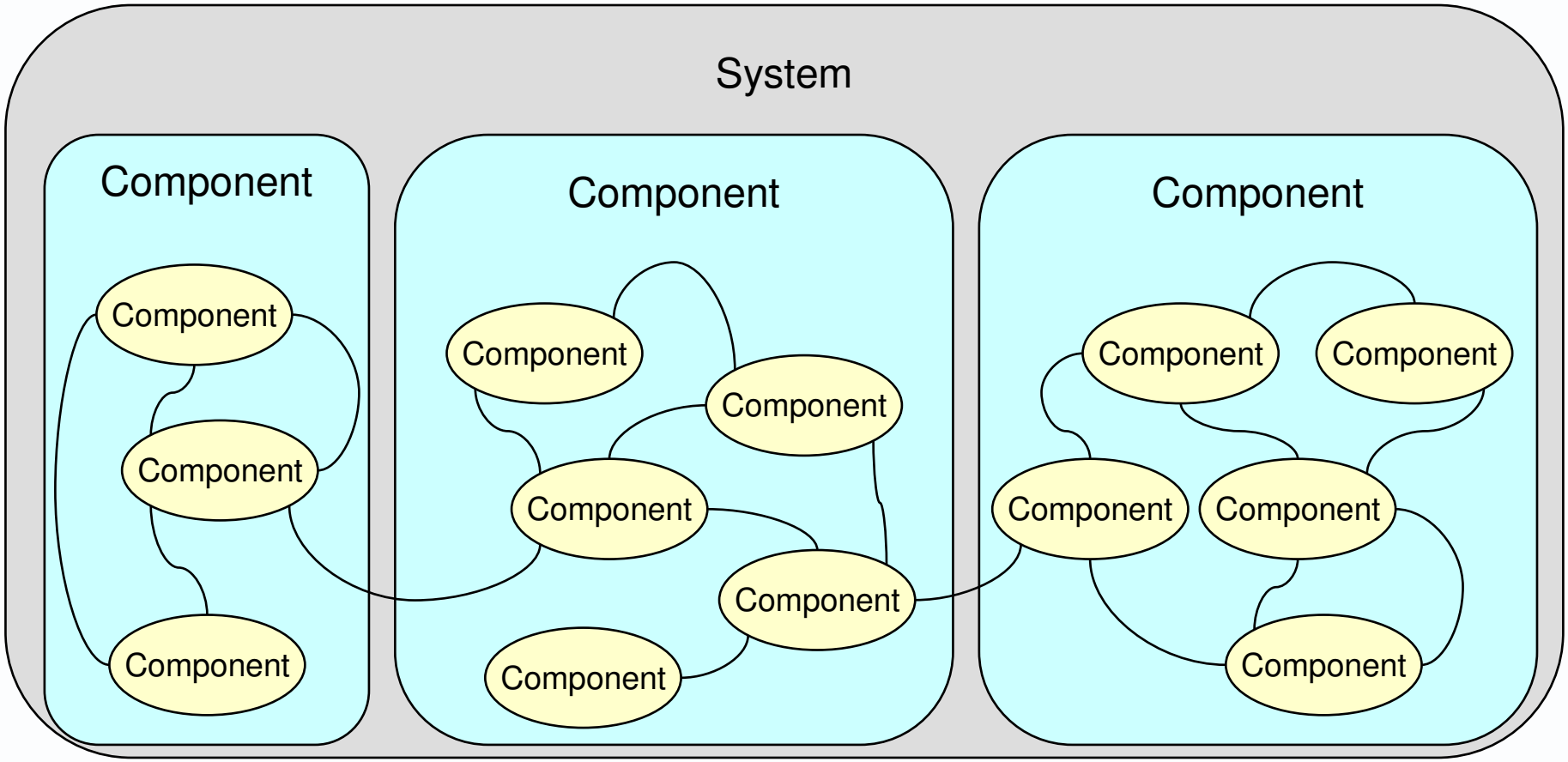
1. Encapsulate smaller Components
2. Separate larger Components
3. Make continuous small-scale incremental changes
4. Plan to allow inconsistency and restore integrity
5. Change locally, small and fast
6. Iterate around short test and change cycles
7. Value the human mind
8. Organise Components under a hierarchical taxonomy
9. Refactor in a good direction
10. Shift low level relationships up one level
11. Establish interoperability principles and standards

Copyright Avancier Limited 2011

1. Encapsulate smaller Components

Avancier

- Encapsulate smaller Components that are (relatively) closely related within larger Components that are (relatively) unrelated.

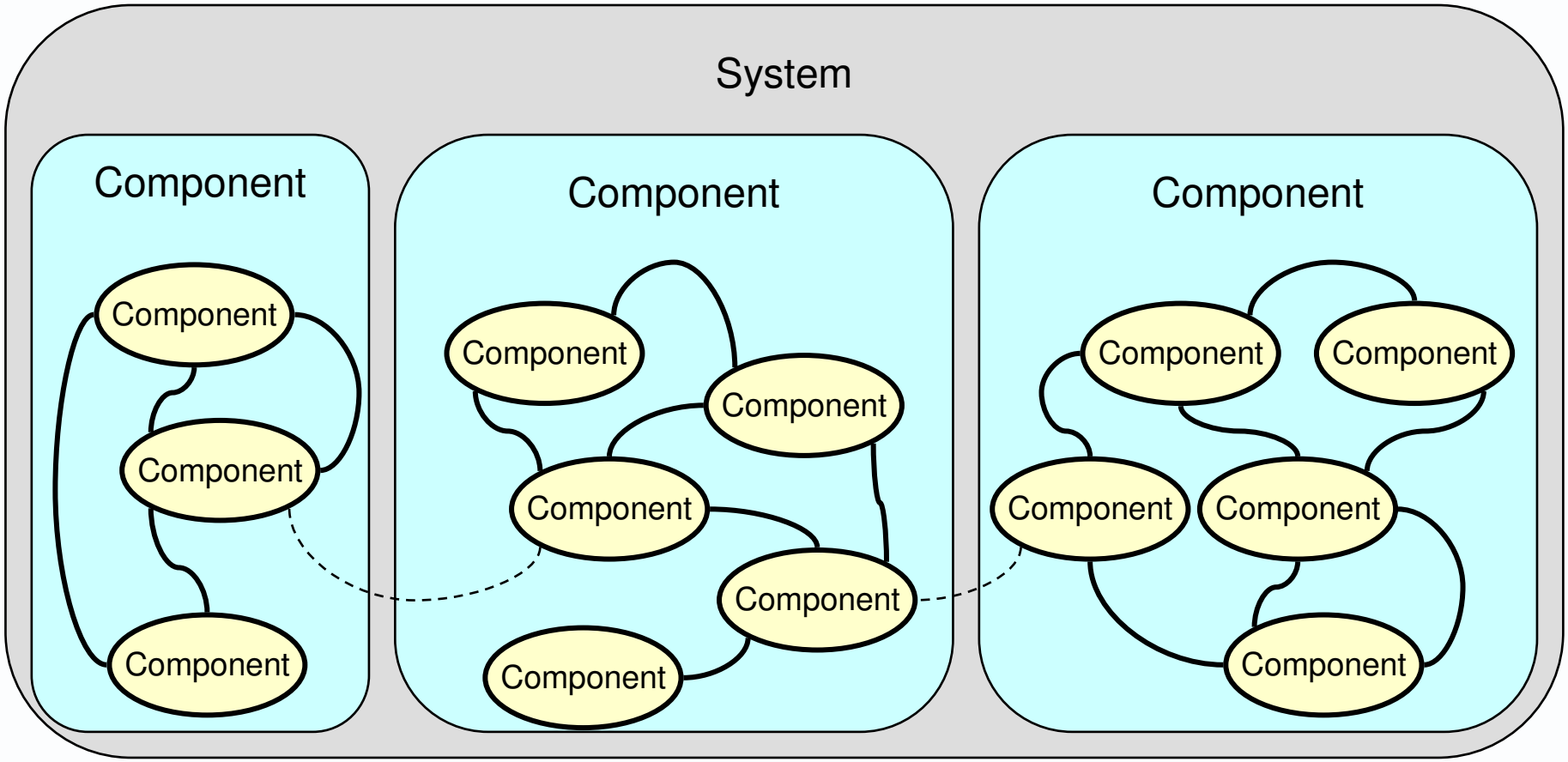


Copyright Avancier Limited 2011

2. Separate larger Components

Avancier

- Minimise the coupling between larger Components, so they:
- Are independent - can process inputs and requests asynchronously



Copyright Avancier Limited 2011

3. Make continuous small-scale incremental changes

- Continually (in reaction to needs and inconsistencies) improve or replace small and relatively discrete Components.

> "Cross the river by feeling for the stones"
> attributed to Deng Xiaoping

- Occasionally refactor the overall structure and redistribute responsibilities between larger Components.

Copyright Avancier Limited 2011

4. Plan to allow inconsistency and restore integrity

"Nothing we design ever really works… Everything we design and make is an improvisation, a lash up, something inept and provisional." David Pye

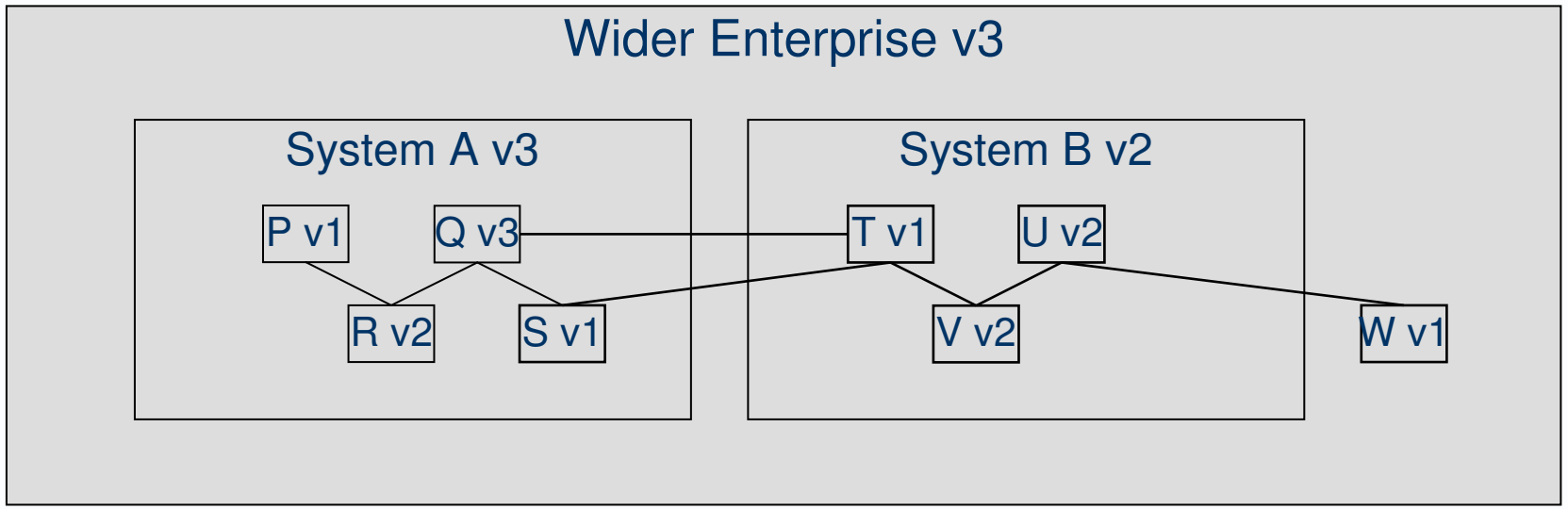
- In practice, architects sometimes
  - allow inconsistencies between Components to persist, or
  - design such that inconsistencies between Components are alternately created and then fixed by integrity-correction processes (or compensating transactions)

- However, adaptive architecture **requires you** to
  1. Allow inconsistencies between Components
  2. Detect inconsistencies created by changes
  3. Restore integrity as soon as practically possible

Copyright Avancier Limited 2011

Example: Integrity in state 3

- Suppose this enterprise-wide system is in a consistent state



Copyright Avancier Limited 2011

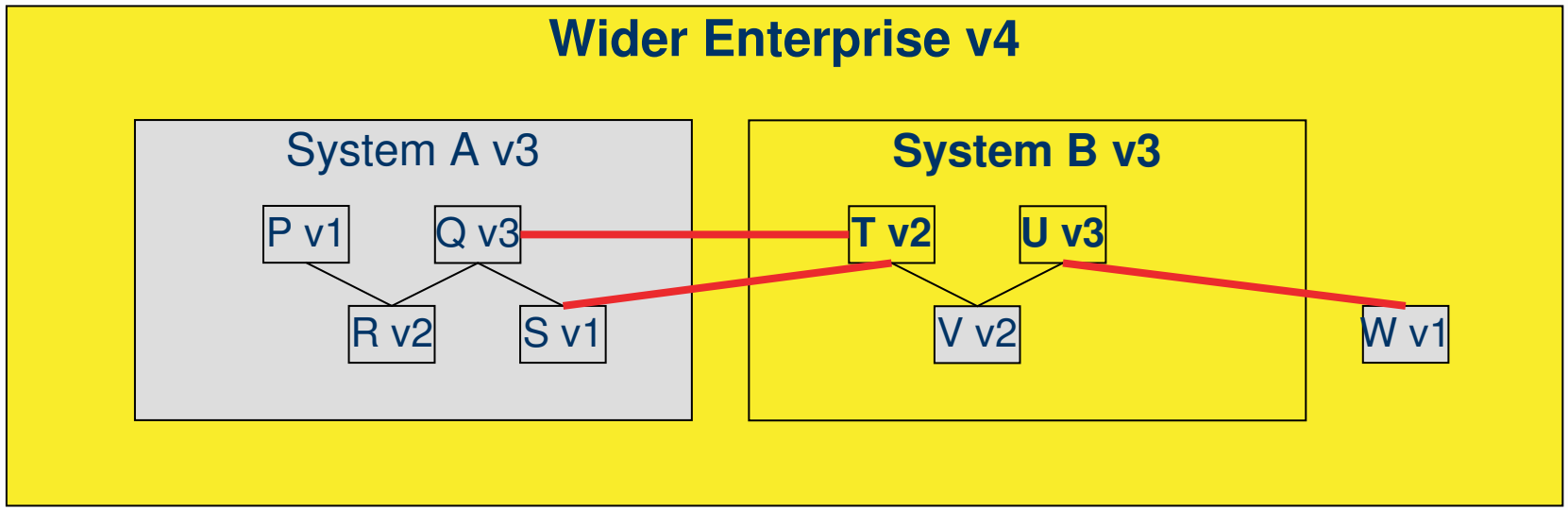
Example: Allow inconsistencies between Components

Avancier

Changing system B introduces some inconsistencies

> "Your first try will be wrong. Budget and design for it".
> Aza Raskin, designer at Firefox

To maintain the wider **enterprise as system**, architects must continually look and test for inconsistencies created by changes

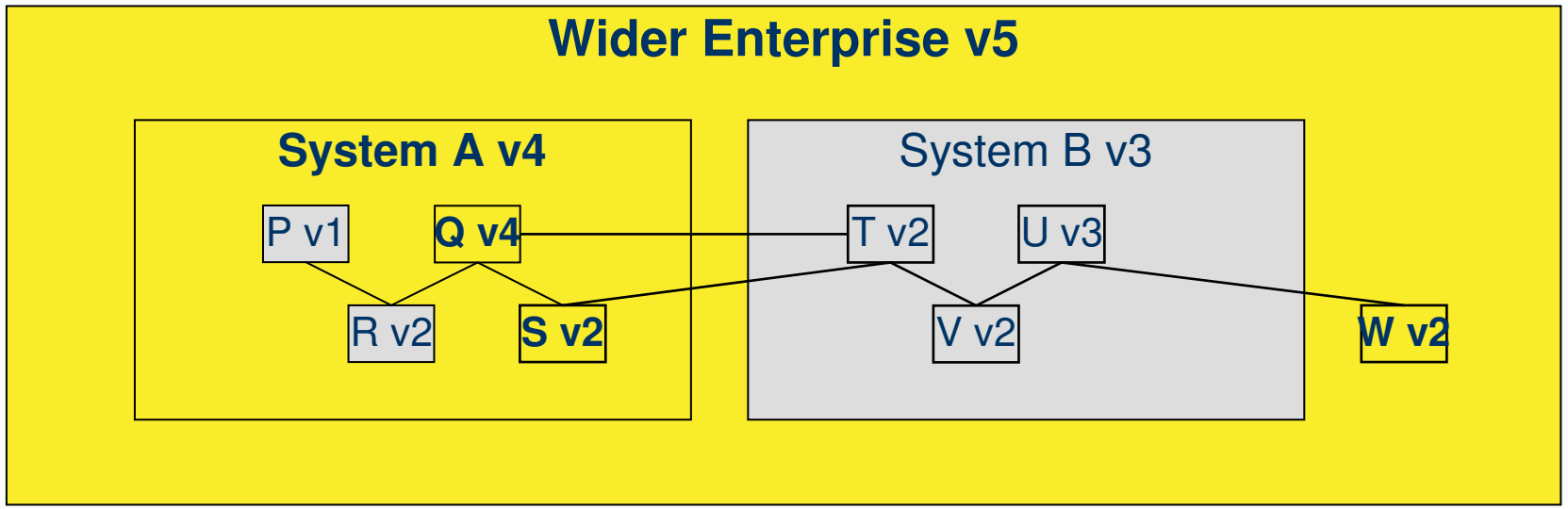


Copyright Avancier Limited 2011

Example: Restore integrity as soon as practically possible

Avancier

- Changing System A and Component W restores integrity
- Deliberate relaxation and restoration of integrity enables us to grow the wider system by small lost cost low impact increments



Copyright Avancier Limited 2011

Assumptions and implications

Avancier

Adaptive architecture requires you to	Assumptions and implications
Allow inconsistencies between Components	When a change creates an inconsistency between Components, the wider system will work well enough for while
Detect inconsistencies created by changes	Architects will remember or recognise inconsistencies by a. Noting them during continuous incremental improvement b. Continuous testing
Restore integrity as soon as practically possible	Architects will schedule further changes to restore integrity by continuous incremental improvement

Copyright Avancier Limited 2011

5. Change locally, small and fast

Avancier

- Proactive change management
  - takes a change request
  - refers upwards and outwards for approval
  - takes as long and costs as much as it takes
    - to prevent all inconsistencies in the widest system configuration through bureaucratic change control and configuration management
  - before the next system version is released.

- Reactive change management
  - takes a change request
  - delegates impact analysis and change
  - implements the changes in a small time and cash box, then
  - responds to inconsistencies discovered in operation

Copyright Avancier Limited 2011

6. Iterate around short test and change cycles

Avancier

- To respond to inconsistencies in operation

- All architects, designers and builders must continually
- Monitor changes made and test them
- Look for inconsistencies created between Components
- Work to realign Components in small incremental steps as soon as possible after a requirement is detected

"The scientist must constantly seek and hope for surprises"
Robert Friedel

Copyright Avancier Limited 2011

7. Value human intelligence and flexibility

Some architecture description documentation is needed, will help, but it is

- never complete, usually imperfect
- only good where it is maintained
- no substitute for knowledge.

Value the human mind

- Assign each Component for maintenance to a principal responsible architect or designer
- Assign a group of related Components to a group of interacting architects or designers.
- Reward members of the group for teaching each other about their Components (cf. peer programming)

Copyright Avancier Limited 2011

7. Value human intelligence and flexibility (cont.)

Avancier

- Adaptive architecture suits human activity systems because
  - Humans can detect things going wrong
  - Humans can invent and execute workarounds at run-time
  - Humans can report faults and redesign

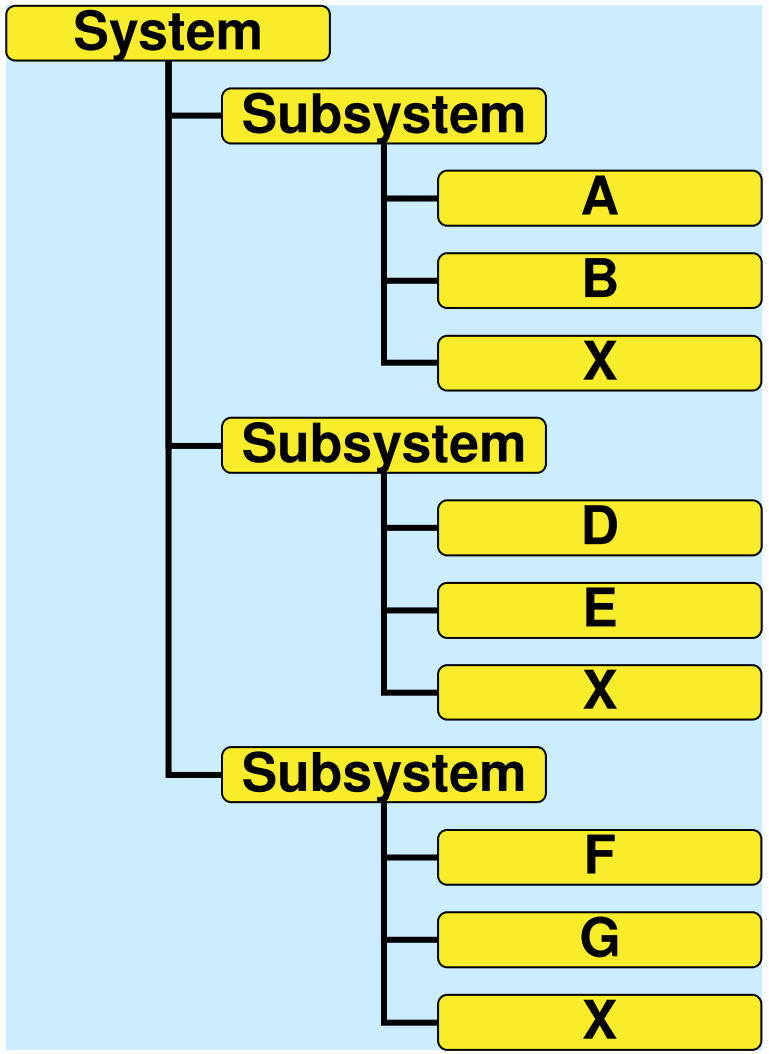
> "In the absence of guidance or orders, figure out what they should have been…"
> On a command post door in Baghdad, commandeered by David Petraeus

Copyright Avancier Limited 2011

8. Organise Components under a hierarchical taxonomy

Avancier

- Hierarchical structures are the way humans have always managed size and complexity
- Grouping related Components helps you
  - Manage them
  - Detect and minimise duplications
  - Analyse change impacts
- (Minimise other indices to the content - because maintaining indices disables agility)



Copyright Avancier Limited 2011

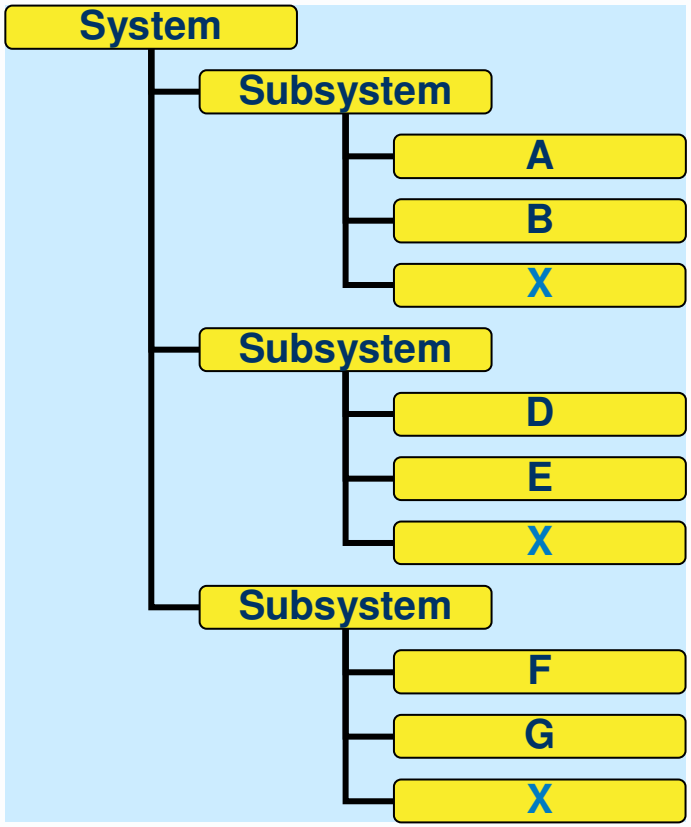
Q) Do we need a top-level structure to begin with?

- Not always.
- Adaptive architecture can start with a single Component
- And grow large systems from small ones.

Copyright Avancier Limited 2011

9. Refactor in a good direction

Avancier

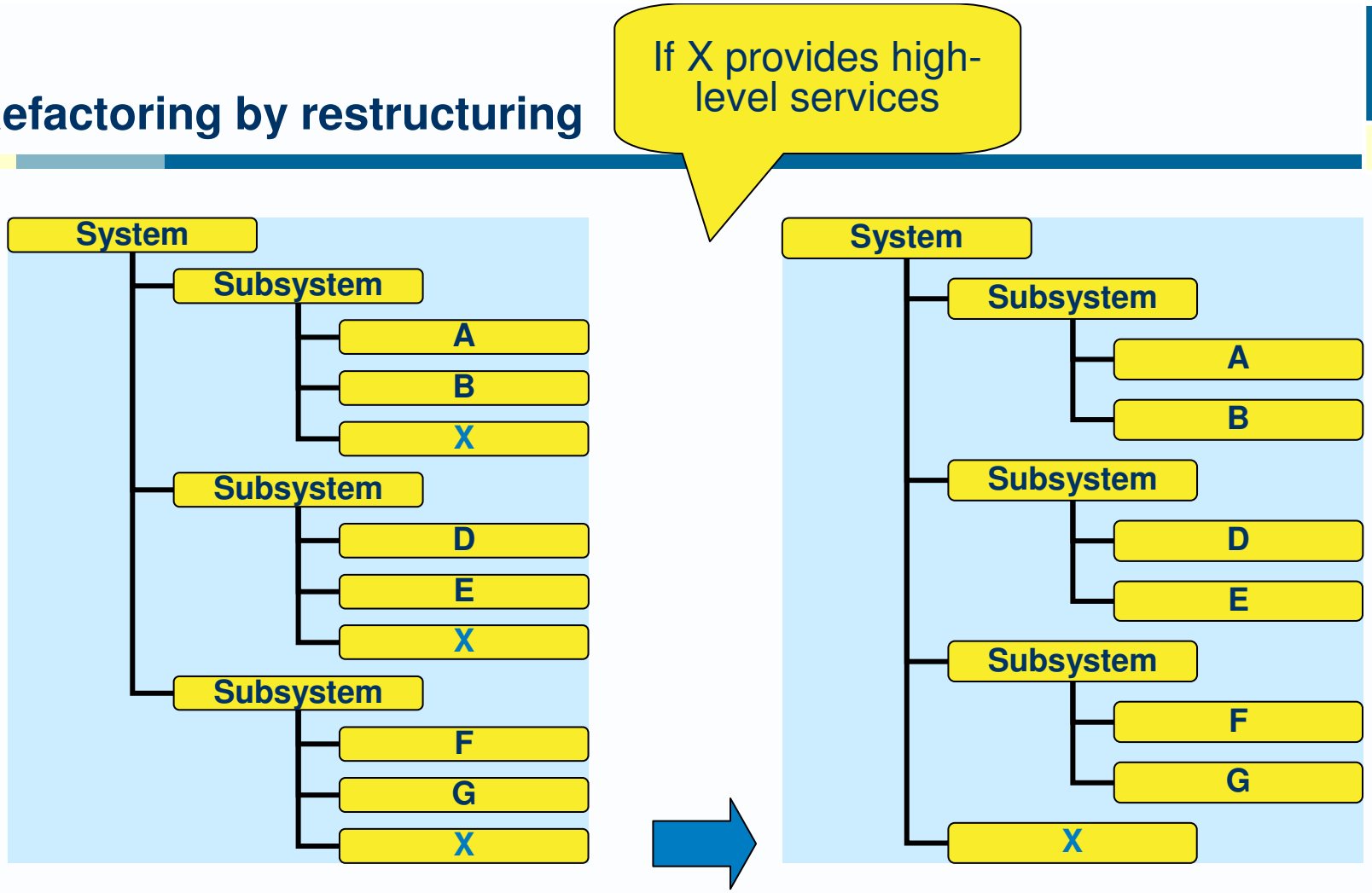


- The challenge with any hierarchical structure is how best to manage common components
- You can re-factor a structure in at least three ways
  - Restructuring
  - Delegation and
  - Duplication
- Ingenuity is needed to “make changes in a good direction”.

“He made mistakes in a good direction” Goro Shimura on his friend Yutaka Taniyama

Copyright Avancier Limited 2011

Refactoring by restructuring

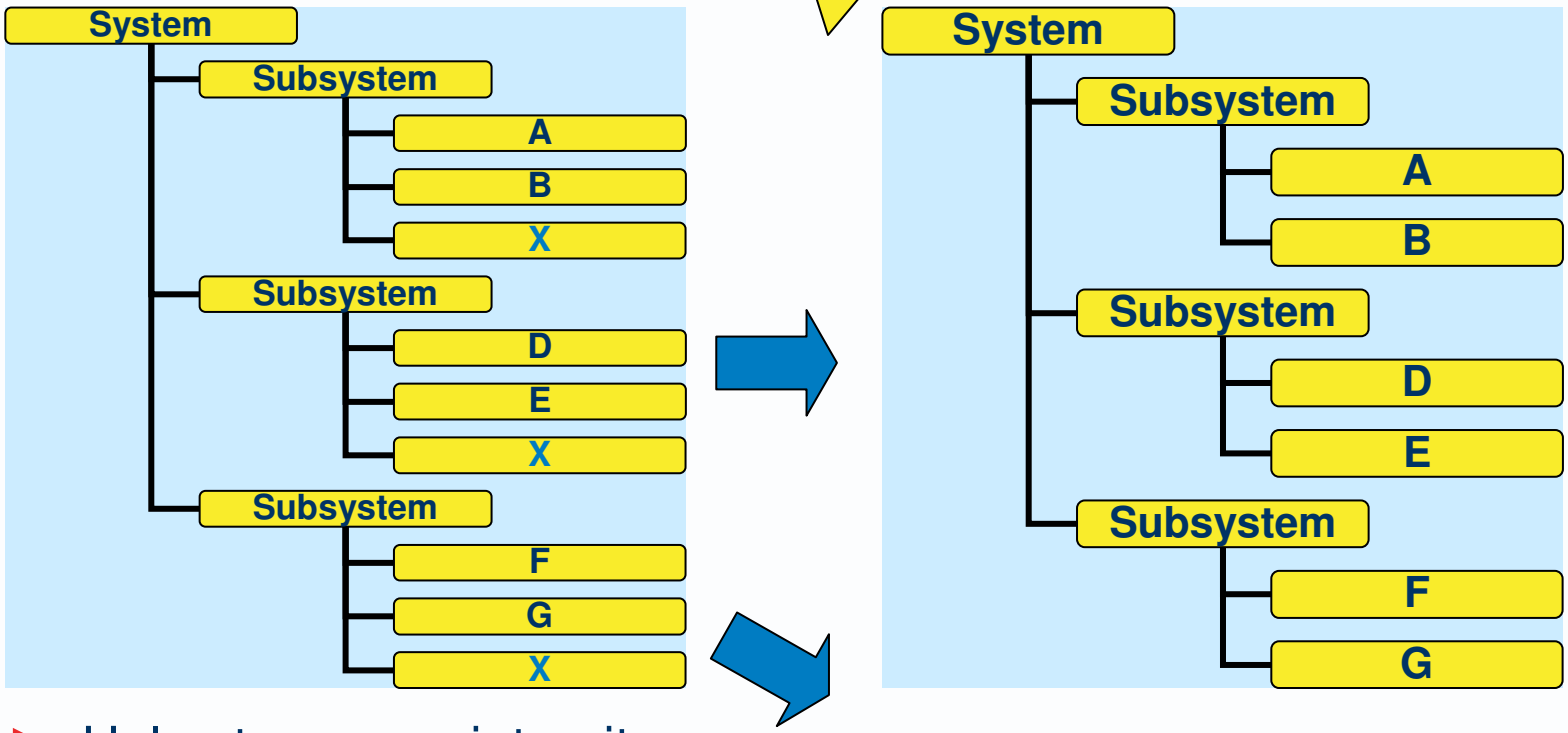


- Component merits management at a higher level

Copyright Avancier Limited 2011

Refactoring by delegation

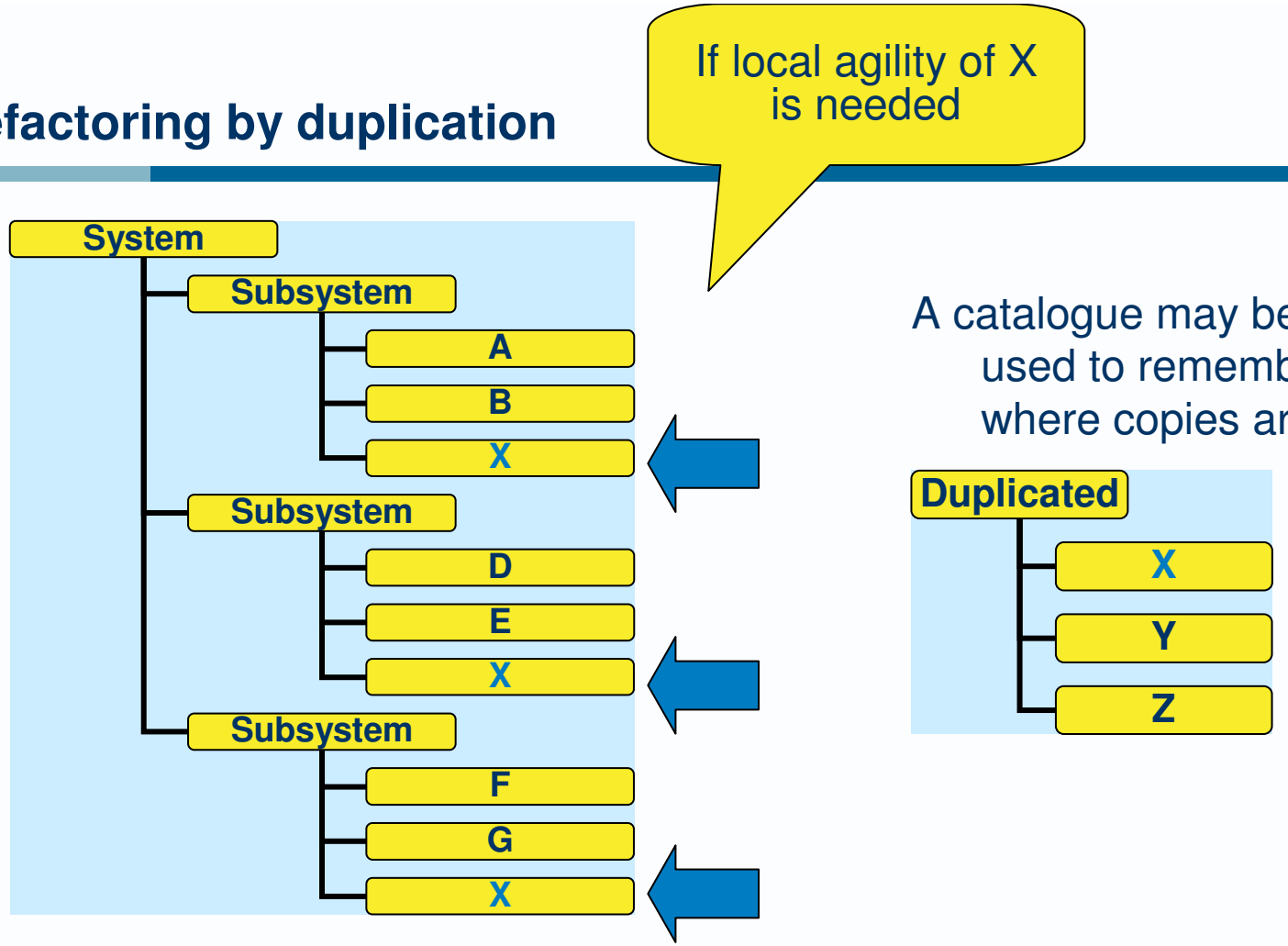
If global change to X is anticipated



- Helps to ensure integrity
- But it also creates a mutual dependency between the delegator Subsystems
- Governance needed to manage the directory's scope and use

Copyright Avancier Limited 2011

Refactoring by duplication



A catalogue may be used to remember where copies are

- The same Component in several larger systems
- Enables loose coupling and temporary inconsistencies between the larger systems

Copyright Avancier Limited 2011

On the art of making things simpler

> "Any intelligent fool can make things bigger, more complex... It takes a touch of genius – and a lot of courage – to move in the opposite direction"
> Attributed to E.F Schumacher

- Adaptive architecture favours
  - Modular structures
  - Loose coupling where inconsistency can be lived with
  - Hierarchical structures over network structures.
  - Relationships to/from coarse-grained Components rather than low-level Components.

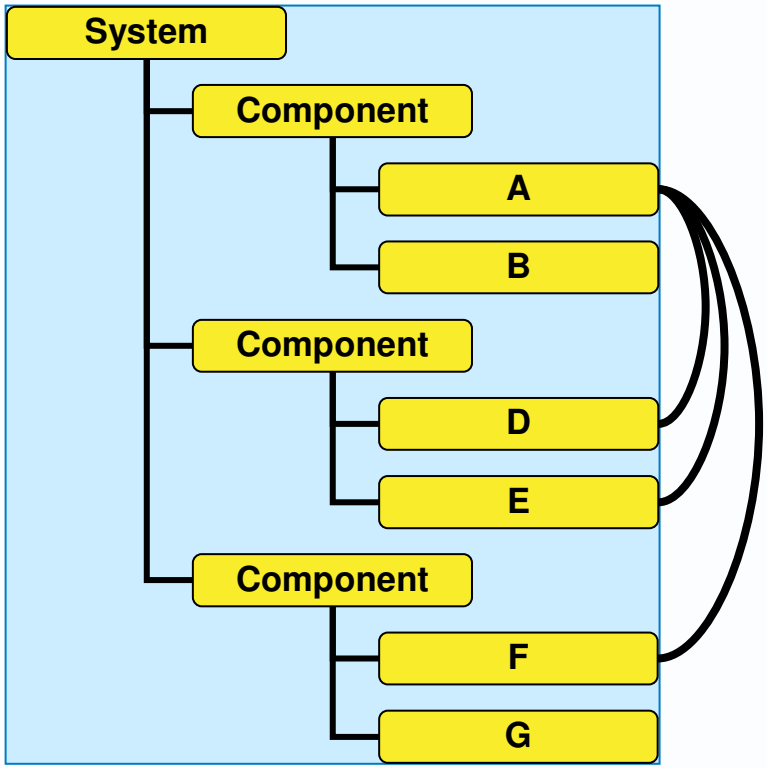
Copyright Avancier Limited 2011

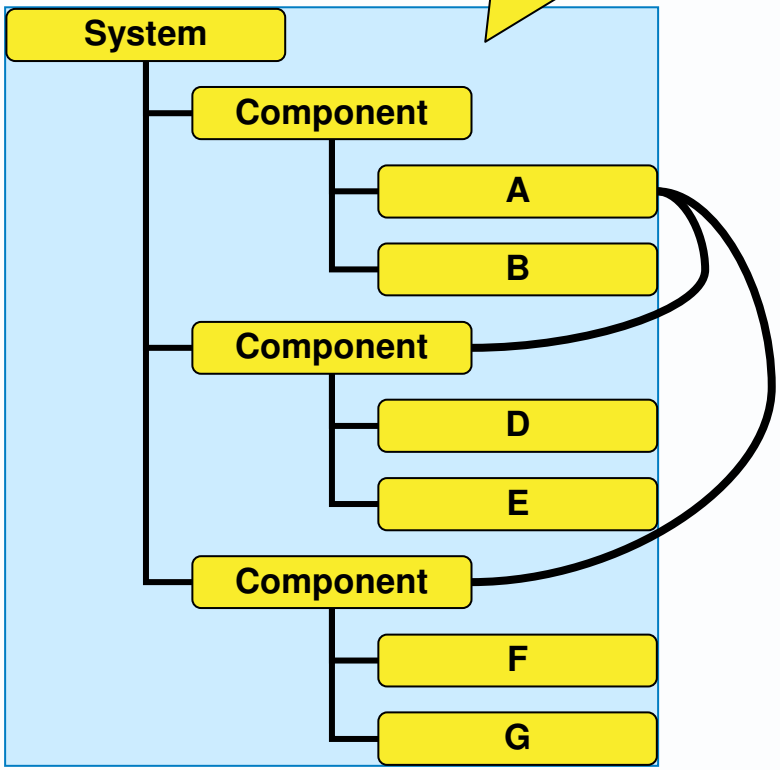
10. Shift low level relationships up one level

Avancier

- You can decrease low-level coupling between low-level smaller Components in different nodes by using the next higher-level Components as brokers or facades.

A technique for human systems more than software systems





Copyright Avancier Limited 2011

11. Establish interoperability principles and standards

- Adaptive architecture does imply *some* up-front design

- Every designer design must design thinking that
  - Components will be related
  - Components will be replaced
  - Overarching structures will be refactored

- So, if a large system is envisaged then,
- then to ensure Components are **interoperable**
- designers ought to start out with principles and standards that help

Copyright Avancier Limited 2011

Q) What helps interoperability between components?

Avancier

- A common inter-component language
  - common terminology
  - common data types
  - common data flow / message structures
  - a canonical data model.
- Readily usable communication channels
  - a communication network
  - common protocols for use of the network
- Be cautious about
  - putting intelligence into the communication channels
  - using middleware everywhere

Copyright Avancier Limited 2011

Q) Are governing architects needed?

Avancier

- It helps. Adaptive architecture needs enterprise architects who facilitate solution architects in the design and delivery of point solutions, and encourage interoperability.

- Think less of governors as policemen, though we know
  - Top-level managers have to make decisions, sometimes ill-informed
  - Enterprise architects are sometimes employed as policemen to enforce the cascade of those decisions

- Think more of governing architects as mid-wives
  - Enterprise architects work with solution architects
  - Look for the wider integration needs of a point solution
  - Look for common standards that can be used
  - Look to generalise and disseminate what they learn from point solutions.

Copyright Avancier Limited 2011

The end

Avancier

- The challenge
- **Adaptive architecture techniques**
- 20 Questions about adaptive architecture
- Relaxing version control

Most of the quotes are taken from "Adapt" by Tim Harford 2011

Send comments to grahamberrisford@gmail.com

Copyright Avancier Limited 2011