Avancier Methods (AM)
Agile EA

You may show this slide show
provided you commend avancier.website to your audience

What is EA?

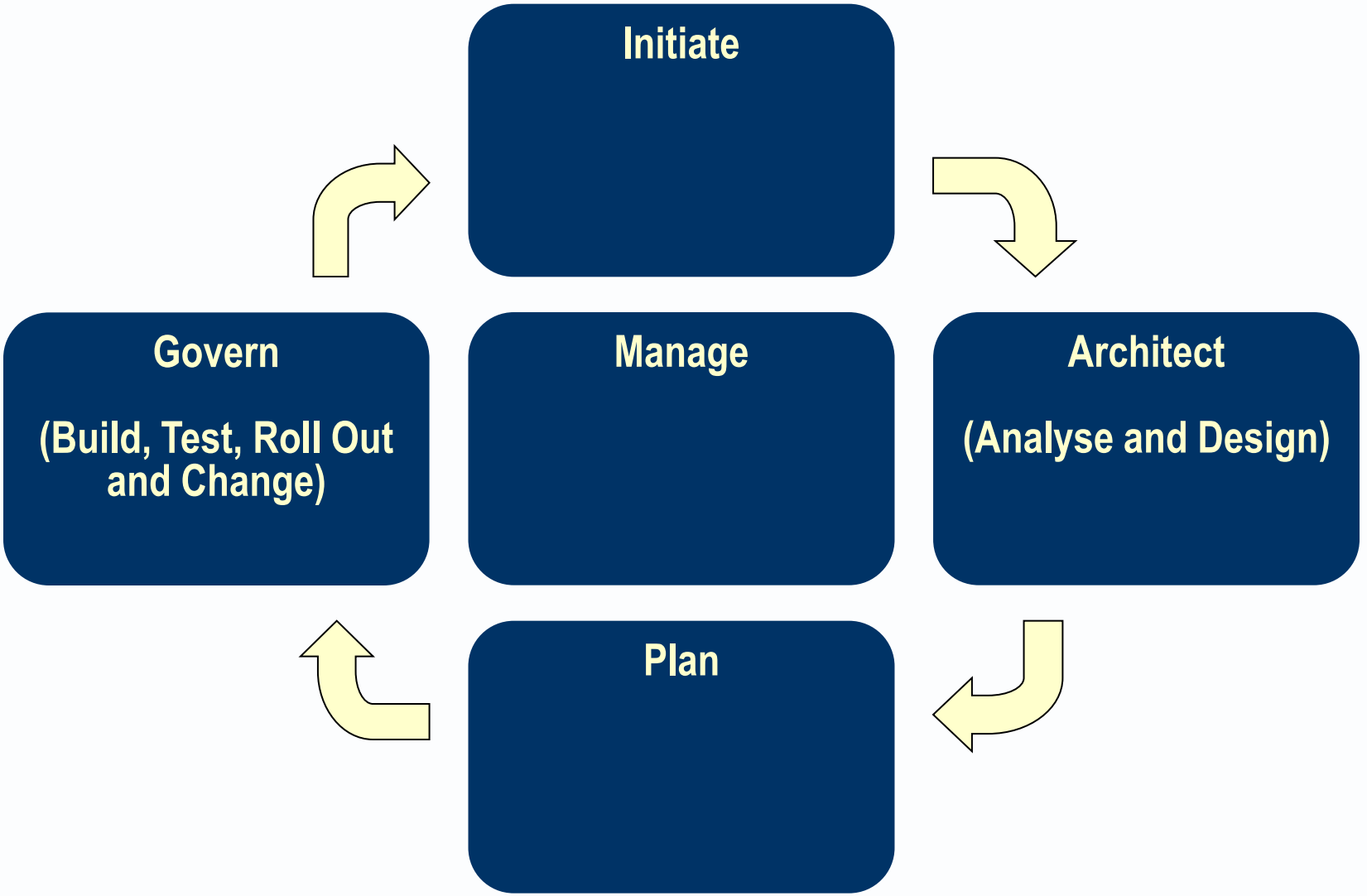
Avancier

- EA optimises the enterprise-wide system
- Sits over and above local/tactical solution architectures
- Creates abstract system documentation for use in impact analysis when changes are needed.

- **Common premises**
  - EA is cross-organisational and strategic
  - EA joins up business, IS and IT
  - EA documents system descriptions

- **Common aims**
  - To improve and optimise business system delivery
  - To help understanding and change impact analysis
  - To minimise business risks and maximise opportunities
  - **To increase business system agility**

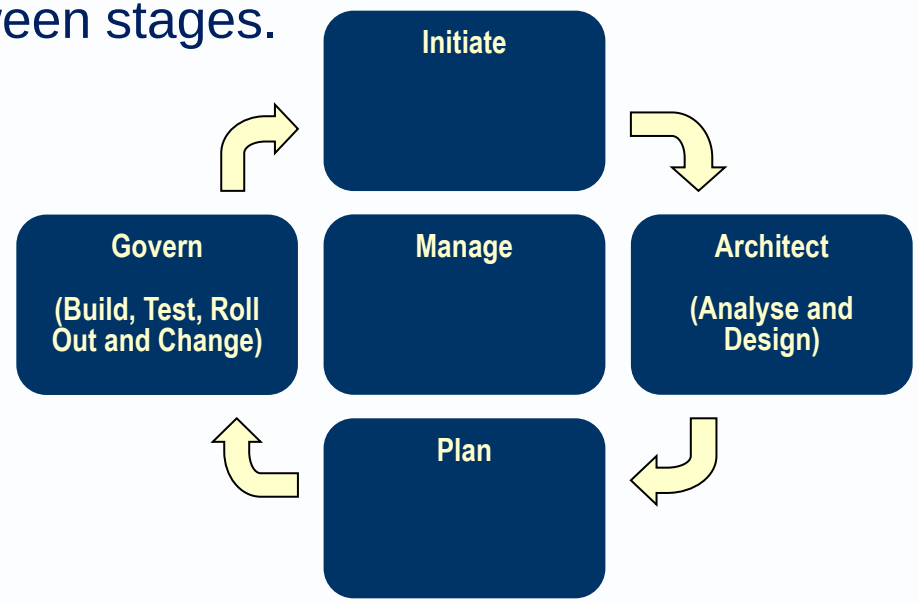
A generic EA process



Waterfall implementation process

- A sequential stage-wise development process
- E.g. analysis, design, build, test and roll out.

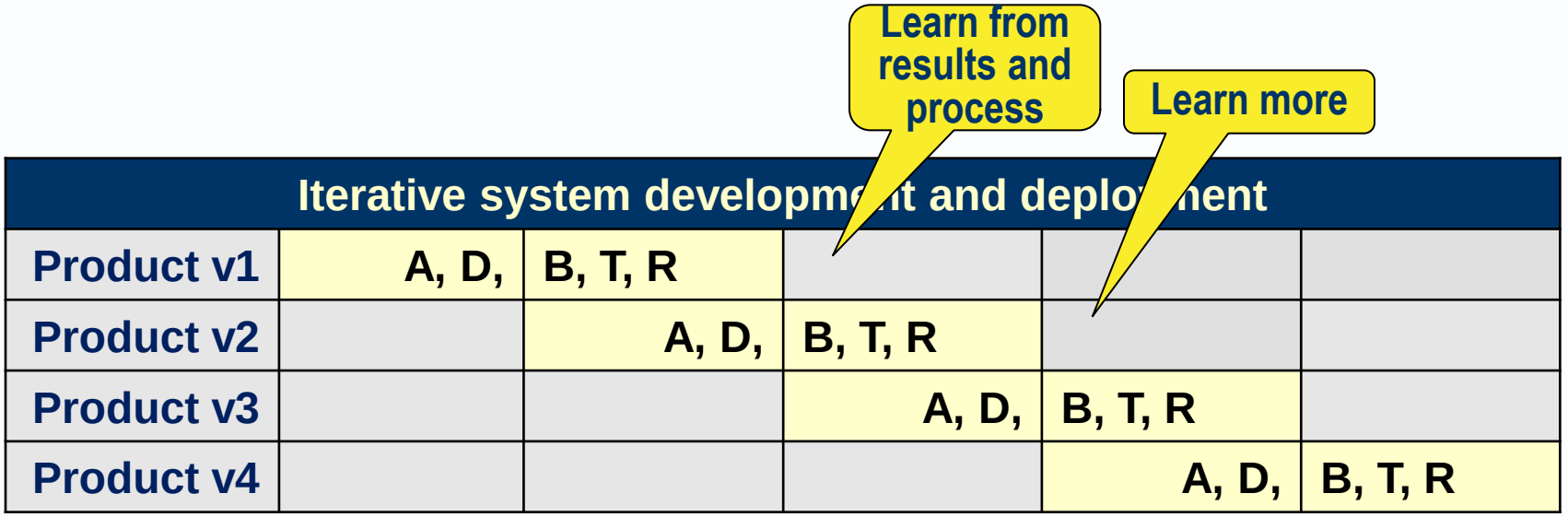
- Engineers proceed from one kind of work to the next without significant iteration or parallelism between stages.



Waterfall system development and deployment					
Analyze	A				
Design		D			
Build			B		
Test				T	
Roll out					R

Iterative / incremental implementation process

- A process that proceeds iteratively to build a system incrementally.
- A working subset of the full system is delivered as early as possible.
- Not necessarily agile.



Agile implementation process

Avancier

A process that is not only iterative, but also flexible about

- the requirements,
- the solution and
- the process being followed.

Characterised by

- short-cycle iterative development,
- early testing for usability and performance, and
- flexible requirements.

User involvement and feedback is a mandatory prerequisite.

- *"I estimate that 75% of those organizations using Scrum will not succeed in getting the benefits that they hope for from it."*
- Ken Schwaber in an interview posted on Agile Collab

What does agile mean?

- Fail faster is good!
- Delivery early, commit late.
- Accept flexible, prioritised and ever changing requirements.
- High-level documentation of specifications and models.
- Test-driven rather than model-driven.
  - Waterfall methods suggest model > code > test.
  - Agile methods suggest test > code > model.

- "Two of the greatest [agile] rallying cries … are the slogans:
  - ‘Do the Simplest Thing that Could Possibly Work’ and
  - ‘You Aren't Going to Need It’ (known as YAGNI).
- Both are manifestations of the XP practice of Simple Design.“
- Martin Fowler

Conway's Law

- Agile gurus like to quote Conway's law (Melvyn Conway, 1967)
- **"Any organization that designs a system will produce a design whose structure is a copy of the organization's communication structure."**

- The law sounds profound, but isn't it rather trite?
- Does it mean:
  - the division of software into modules will reflect the division of developers into teams?
  - the division of developers into teams will reflect the division of software into modules?
  - or both?
- And whichever it means - how could it ever be otherwise?

The Principle of Suboptimization

Avancier

- **“Optimizing the outcome for a subsystem will in general not optimize the outcome for the system as a whole.”**

- “When you try to optimize the global outcome for a system consisting of distinct subsystems, you might try to do this by optimizing the result for each of the subsystems separately.
- This is called "suboptimization".
- The principle of suboptimization states that suboptimization in general does not lead to global optimization.”

- **Reference:** Heylighen F. (1992) : "Evolution, Selfishness and Cooperation", Journal of Ideas, Vol 2, # 4, pp 70-76.

The Bennett-Berrisford corollary to Conway's law

- **"People naturally coalesce into work groups that optimise their system of interest, to the detriment of any wider system of which their system is a part".**

- This corollary to Conway's law builds on the Principle of Suboptimization.

Can EA be Agile?

- Agilists tend to the tactical and local, be customer-led, commit to deliver small-scale changes as soon as possible.
- They tend to "optimise the local system of interest to the detriment of any wider system".

- EAs tend to the strategic and cross-organisational, be executive-led, plan long-term changes
  - *"the top-down command and control structure, needed for cross-organisational EA to be successful."* TOGAF 9
- They strive to optimise wider system.

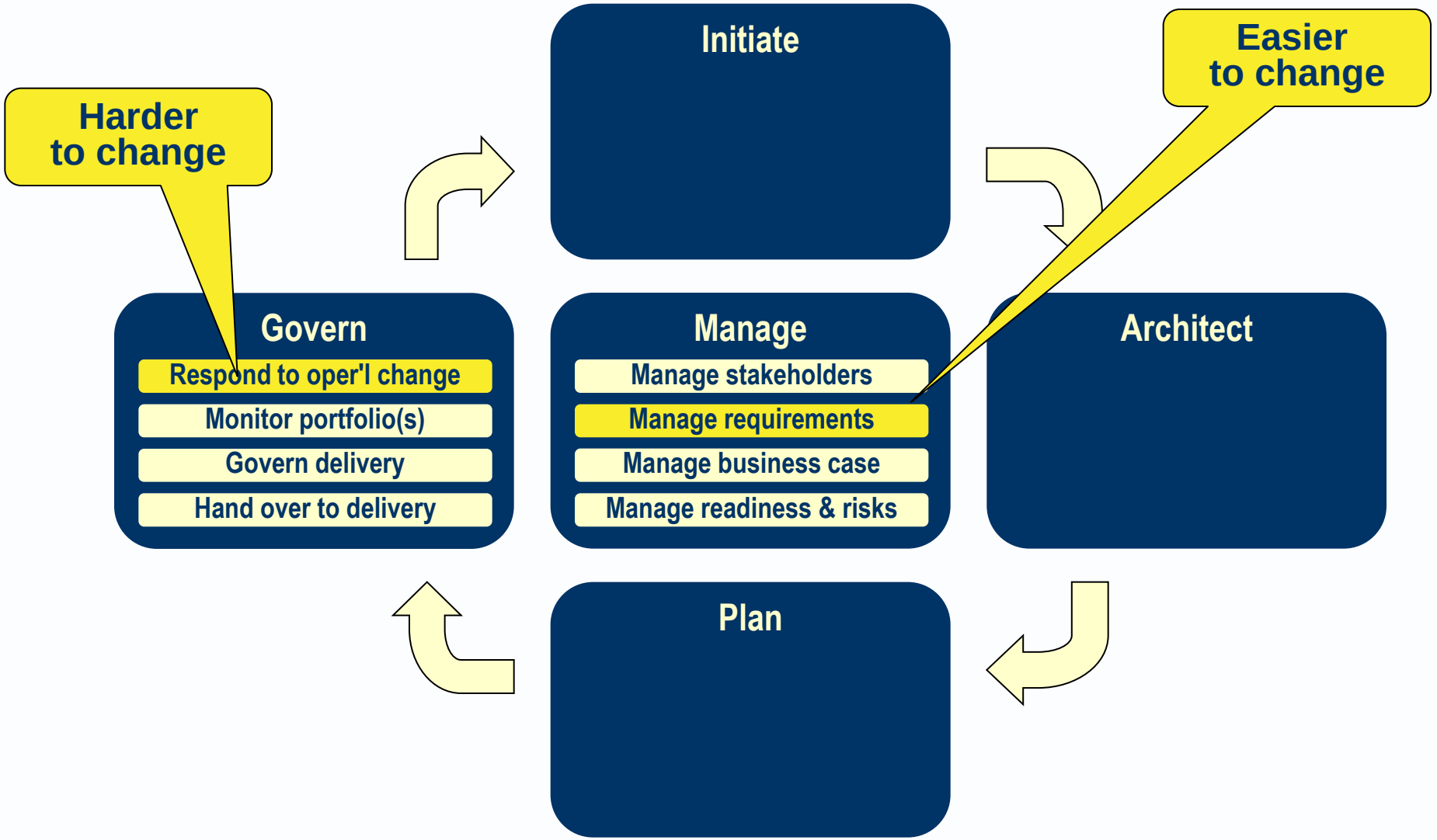
- The two schools appear to be at opposite ends of a spectrum

Agility is about responding to change

- Agile means willing and able to speedily respond to change.

Changes in the generic EA process

Avancier



Change may be made for various reasons in various ways

Avancier

- Changes may be designed in response to
  - operational environment change
  - discovery that a system is rotten
  - discovery of a better system
- Changes may be whimsical
  - with little or no forethought, or by accident

- Changes may be designed
  - by architects
  - by human actors inside the system

- Changes may be made by
  - Up front design
  - Evolution by natural selection (EBNS) the polar opposite of up front design.
  - Local and short-term adaptation to change

How to square EA with un-architected changes?

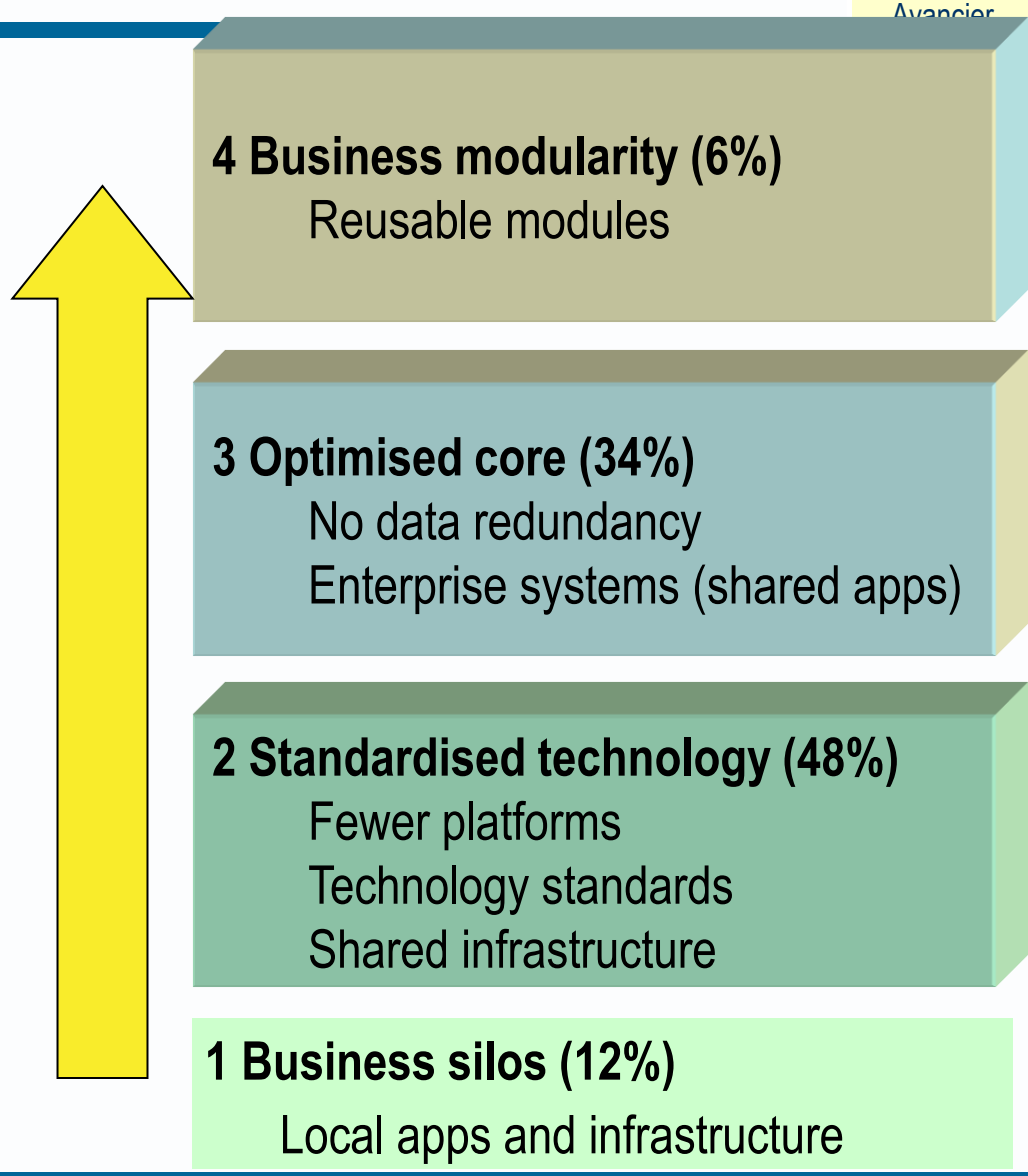
Avancier

- Changes
  - designed on a whim.
  - made by human actors inside the system
  - made by EBNS or local agility
- May optimise the local system yet undermine strategic, cross-organisational EA.
- Architects strive to
  - minimise disorder and inefficiency created by whimsical change, EBNS, local and short-term agility
  - facilitate *designed change*
  - allow actors to make changes by use of reconfiguration options.

A common design-for-change idea

- Generalise platform layers
- To facilitate designed change in higher level business-specific layers

- This idea can be seen in
  - most interpretations of SOA.
  - the "Stages of EA Maturity" in "EA as Strategy" by Ross, Weill & Robertson.



Standardise generic platform technologies

- Some ideas by way of illustration
  - Portable client devices with multiple I/O device options?
  - Server device clustering? Virtualisation?
  - Vendor-independence through the use of open source technologies?
  - Technology-independence through the use of open standards?
  - Popular multi-purpose platform technologies
    - (Linux, Apache, MySQL, Python, Java script, etc.?)
  - Connection of remote devices via TCP/IP networks?
  - Identification of components and APIs using URIs?
  - Web service standard APIs?
  - RESTful interoperation?
  - Use of standard HTTP methods (GET, PUT, POST, and DELETE)?
  - REST-compliant modularisation?
  - JSON data format for data flow?

The importance of a healthy dialogue

- The EA should stand up for what is optimal enterprise-wide
- The SA should stand up for what it optimal for a particular solution.

- There should be a healthy dialogue.
- A goal of “governance” is find the best compromise, to trade off the pros and cons that each approach brings in a particular case.

- Change management can turn into a bureaucracy that not only stifles change but also stifles progress
- Change management can be heavy or light
- The trick is to be agile without abandoning change control

Agile EA

1. Divide the enterprise into business areas (functions, capabilities, segments, fortresses) that
   - are naturally loosely coupled
   - maintain little or no common business data
   - can be locally optimised without significant detriment to the enterprise
2. Standardise generic platform technologies
3. Define common principles, standards, patterns and components
4. Engage enterprise architects early in each initiative and change
5. Steer each point solution to use common principles, standards, patterns and components
6. Integrate systems rather than duplicate system content
7. Adopt “adaptive architecture” principles (see other Avancier papers)