Avancier

Avancier Methods (AM)

There is an address for comments at the end

Adaptive Architecture

A first draft or manifesto v8

Contents

Avancier

A common critique of Enterprise Architecture (EA)

- High level managers and enterprise architects
  - are out of touch.
  - promote vacuous generic principles regardless of the variability of situations on the ground
  - make big and costly decisions with too little information
  - kick off large-scale transformations that cannot be completed
  - leave us intermediate or transition states that screw things up by being more complex than either baseline or target state
- Top down command and control doesn’t work

Avancier

Two responses to the common critique

- “Guerrilla EA”
  - Iterative solution architecture, done with a strategic mind set.

- “Adaptive EA”
  - continuous incremental improvement that
  - allows short-term dis-integrity, but always with
  - a plan to restore integrity later.

Enterprises as systems

- “Enterprise architecture structures the business planning
- into an integrated framework that
- regards the **enterprise as a system** or system of systems.”

TOGAF 9.1

Most later quotes are taken from “Adapt” by Tim Harford 2011

Interconnectivity of system elements

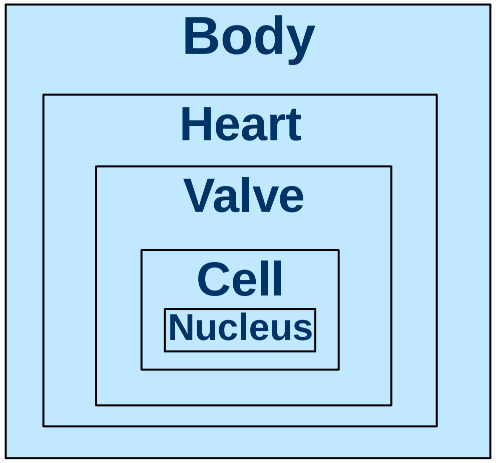
Avancier

- A system is a collection of interrelated elements.
- A system's elements must
  - be related directly or indirectly,
  - form a coherent entity
- else there would be two or more distinct systems
  - or silos as they are sometimes called

EA is concerned with system integration: with cross-organisational integration of systems, and integration of business and IT systems.

Two kinds of element

- **Structural** decomposition identifies **subsystem** elements
  - Reduces a body to components or organs like
    - lungs, heart
    - liver, kidneys
    - stomach, small intestine, colon,
    - brain, nervous system
    - etc.

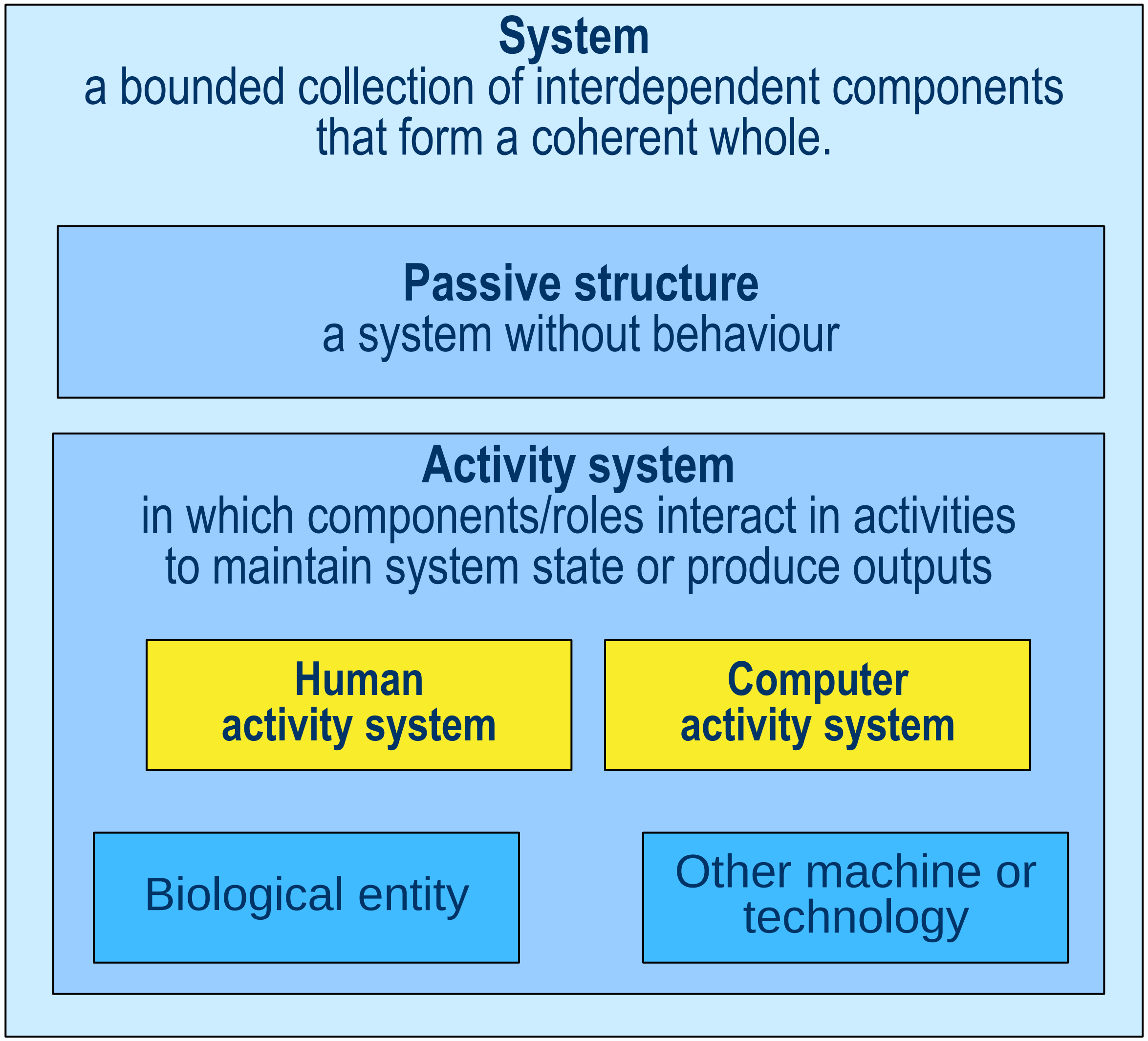


- **Behavioural** decomposition identifies **process** elements
  - Reduces a body to processes like
    - breathing, respiration, perspiration,
    - ingestion, digestion, peristalsis,
    - sexual activity, sleeping, expectoration,
    - etc.

Activity systems

- Many definitions of "architecture" emphasise structure.
- A passive system has a structure, but no behaviour.
  - A Greek temple
  - The Dewey Decimal Classification system
  - The Zachman Framework
- But EA focuses on **human and computer *activity* systems**
- Featuring human roles and software components

Activity system types

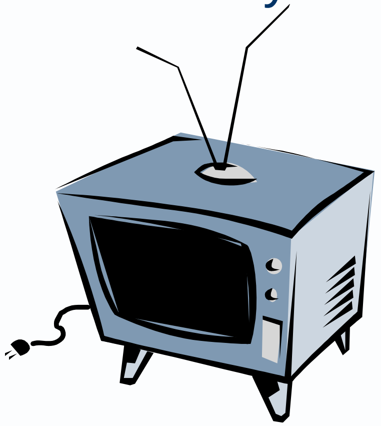

System
a bounded collection of interdependent components that form a coherent whole.
Passive structure
a system without behaviour
Activity system
in which components/roles interact in activities to maintain system state or produce outputs
Human activity system
Computer activity system
Biological entity
Other machine or technology

- Enterprise architecture (EA) and solution architecture (SA) are focused on
  - enabling and improving business roles and processes that are
  - deterministic enough to be
  - systematised and digitised.
- So, enterprise and solution architects are expected to build models of human and computer activity systems.

Architecting and change

Avancier

- The architecting of a bridge, car or television is usually a one-off exercise – the structure and behaviour of the built system are relatively stable.



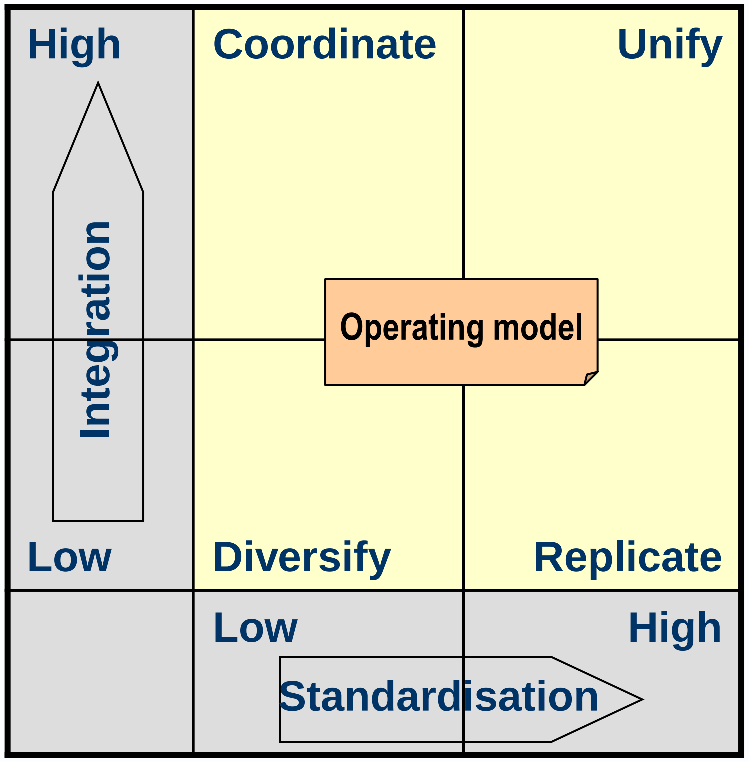
- The architecting of a human and/or computer activity system involves not only
- design *of a* change but also
- design *for* change.

EA is not only about design *of a* change

- Given "EA as a Strategy"

- Many would say the role of EA is to
  - **transform an enterprise**
  - by increasing the
  - cross-organisational integration and/or
  - cross-organisational standardisation of
  - business processes and
  - the systems that support them.

- Why?



EA is also about design *for* change

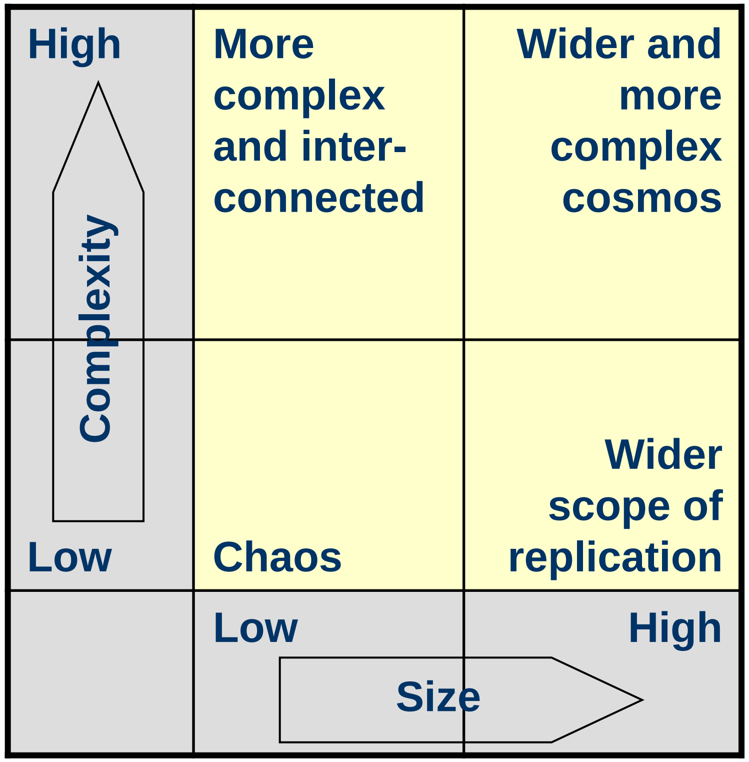
- EA is not only to improve current business-IT alignment
  - facilitate understanding of the business
  - support business processes effectively and efficiently
  - maximise the value of IS and IT and minimise its cost

- But also to improve future **agility,** to improve
  - top-down direction of the business
  - change and change management
  - business and technical agility
    - IS and IT should be changeable to match business change
  - the speed, cost and quality of future solution delivery

- *Few speak of the prices to be paid for these benefits!*

The price paid for EA as Strategy

- Cross-organisational standardisation - **increases the size**, scope and volume of the system we are trying to manage.
- Cross-organisational integration - **increases the complexity** of the enterprise or wider system we are trying to manage
- In other words, EA naturally tends to **increase size and complexity of the system(s)** that are architected and managed
- And this has an impact on change management.


High
Complexity
Low
More complex and inter-connected
Wider and more complex cosmos
Chaos
Wider scope of replication
Low
High
Size

The barrier to agility

Avancier

“The barrier to change is not too little caring, it is too much complexity”

Bill Gates

In theory (according to our quality managers)

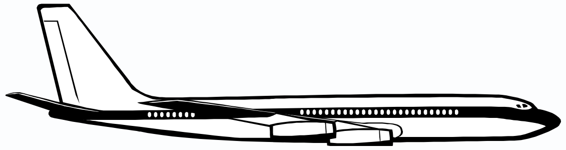
Avancier

- Change management is a process that
  - takes a change request and
  - prevents all inconsistencies in a system configuration through
  - change control and configuration management
  - before the next system version is released.

Top down change control

- The organisation, processes and documentation you need to
- ensure a system is **perfect** before use
- change a large and complex system **safely**

- E.g.
  - an airplane
  - a railway network
  - the cash machines of a bank.
  - the configuration of any software configuration whose integrity is vital, whose results must be perfect.



In practice

- As the size and complexity of a system increases
- So change management tends to become *disproportionately*
  - Slower
  - More costly
  - More bureaucratic
  - More likely to fail

"It's so damn complex. If you ever think you have the solution to this, you're wrong and dangerous."

H R MacMaster

Of course

- Man-made systems cannot evolve without any up-front design
- Executive-level forces demand executive-level strategies and top-down direction
- Dependencies between systems can be deeply buried or hidden
- Documentation will always be needed to
  - explain complex systems, and
  - direct attention to where change impacts might be

- And yet

Avancier

And yet… we have to admit

- Systems grow too large or wide to be changed all at once
- Systems grow too complex to be changed without error

- Increased system integration spreads change impacts
- Change requests arrive faster than top-down change control can process them
- No person or team understands the whole system - many people must be consulted – just in case
- Documentation is never enough for full change impact analysis because it takes too much time and cost to
  - Document millions of components and inter-dependencies in enough detail
  - Maintain the documentation in the light of system changes
  - Read and interpret what is inherently abstract documentation

30 years of software development history taught us that

- As software systems grower larger and more complex
- increasing the emphasis on
  - up-front design and
  - top-down change control
- shows diminishing returns and eventually hinders progress

- Other approaches have evolved to ensure the quality of systems delivered
  - Agile development (short-cycle dev-test cycles)
  - Standard solution design patterns

Agile development

- Seeking ever more input from more stakeholders by way of requirements specification can lead to paralysis by analysis
- Some requirements (e.g. functional and usability) are better addressed well through short dev-test cycles.

Agile	Willing and able to speedily respond to change.

- Other agile principles and disciplines include
  - Fail faster is good
  - Changes are welcome
  - Short-cycle iterative dev-test cycles
  - Tests are executable specifications
  - Minimal documentation until the system has stabilised
  - Team knowledge sharing
  - Reverse engineer specifications as needed

Solution design patterns

- Some requirements, especially non-functionals such as throughput, availability, recoverability, security do need to be addressed up front.
- If we can classify the system NFRs, and then map it to the corresponding solution design pattern(s), we can greatly cut down both:
  - the need to elicit and reconcile different stakeholder views
  - the amount of new architecture description required.

So what can enterprise architects do?

- We can develop a large and complex enterprise or system quickly and cheaply
- If we are willing to pay the prices for that
- Among other things, agility implies if not demands
  - Division of an enterprise into loosely-coupled systems
  - Division of a system into loosely-coupled components
  - A willingness to change to one system/component at a time and tolerate inter-system/component inconsistencies *for a while*

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