AARHUS UNIVERSITET

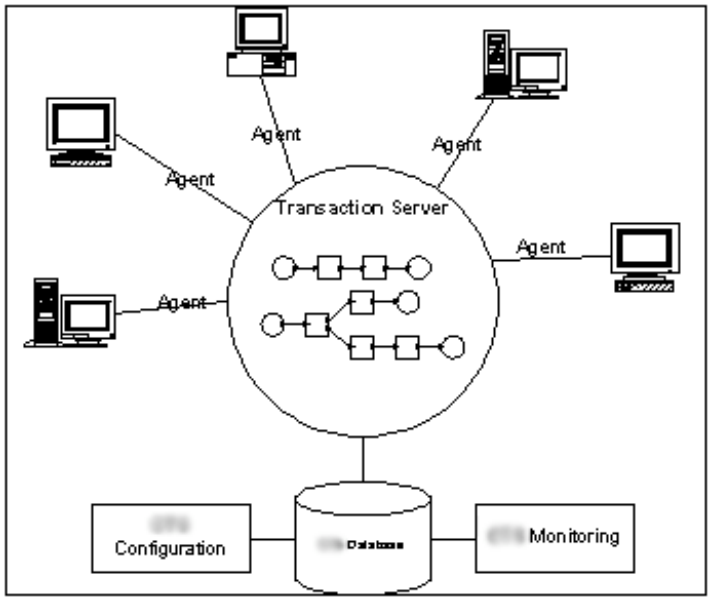
Software Architecture in Practice

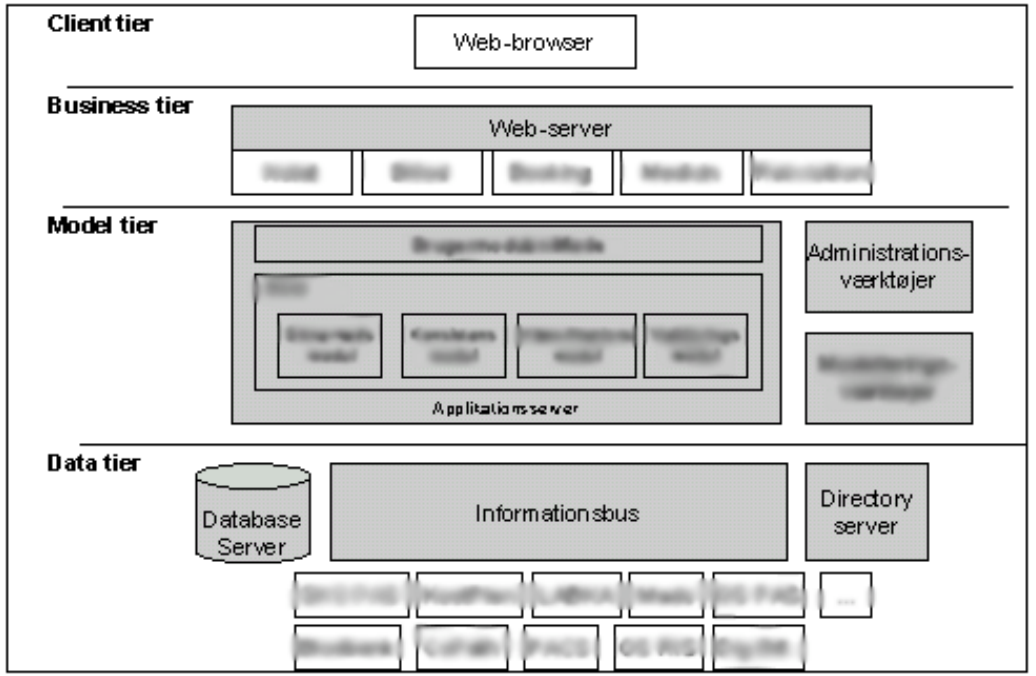
Language supports communication!

Definition

Views

Software Architecture?



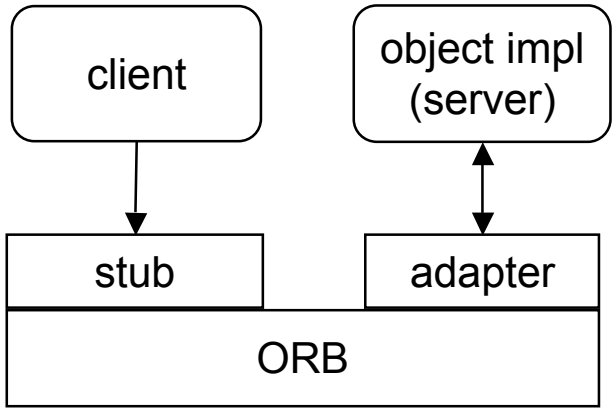


What is Software Architecture?

AARHUS UNIVERSITET

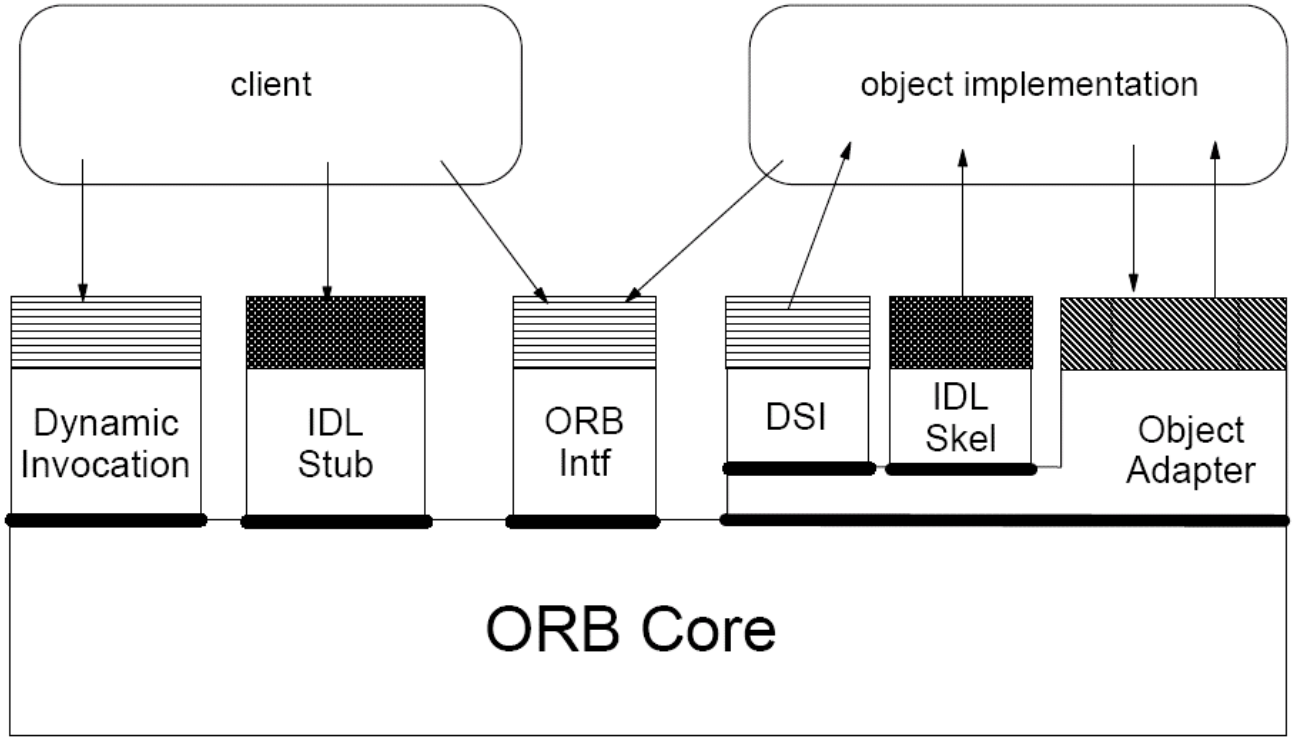
The architecture of CORBA:

Common Object Request Broker Architecture



?

A Better Description?



Box-n-line drawings

Hence, most of the box-and-line drawings that are passed off as architectures are in fact not architectures at all. They are simply box-and-line drawings.

[Bass et al.]

What is Software Architecture?

”Architecture is the overall structure of the system”

– But what structure?
  - Classes?
  - Run-time objects?
  - Processes/threads?
  - Dataflow?

A definition

The software architecture of a computing system is the structures† of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them. *[Bass et al. 2003]*

- Note: *structure***s**
- externally visible properties:
  - assumptions on individual elements
  - services, performance, fault handling, shared resource usage, security, etc.

† '*views*' is another term that is more commonly used...

Implications

architecture defines **elements**

- architectures are *abstractions*
  - components have public and private parts
  - architectures are only concerned with the public parts

elements have **relationships**

- decomposition require the parts/components be related
  - control flow / data flow
  - dependencies
  - and many others

Implications

architectures comprise more than one **view**

- what are the elements?
  - they can be ”anything”:
  - binary components, processes, modules, programs, libraries, hardware nodes,...

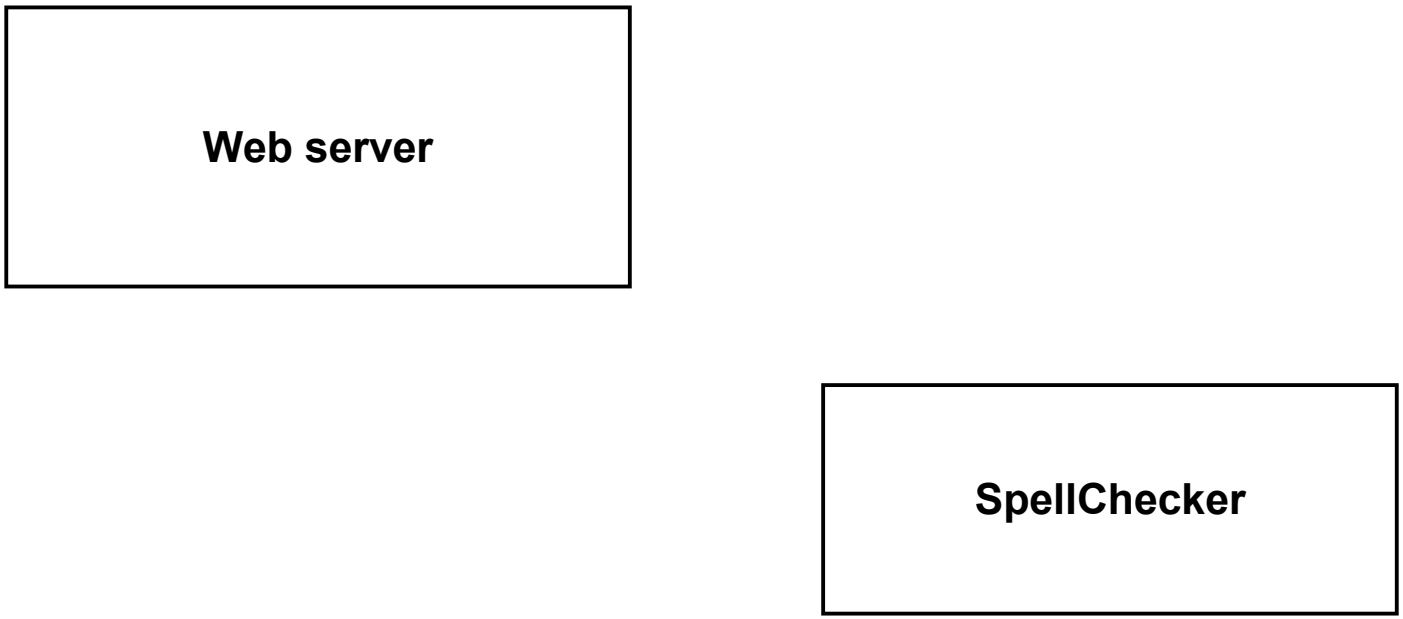
external visible properties of each elements is part of architecture

- interaction patterns, protocols, functionality
- but also: security aspects, performance, …

Exercise

Two components –

- What are the important external visible properties



Your experiences?

What ‘definition’ do you have of software architecture in your company?

Discussion

Every software system has an architecture

- it may, however, not be known to anyone ☺
- meaning there is a difference between *architecture* and *architecture representation*

Architecture:

- *reality* embedded in system

Architecture representation:

- our description of it
  - in documents, in our collective minds, on napkins, …

Discussion

architecture may be ”good” or ”bad”

- raises the issue of *architecture evaluation*
- … and evaluation criteria
- … and who defines the criteria?
- … and how we measure?

How do you define ‘good’ ?

How do you measure quality?

Why SA?

Software architecture is an investment

– we invest in it in order to reduce costs in the software production...

Software Architecture as **Abstraction** !

– omit detail in order to cope with complexity
– common overview – common goal
– overview and compact documentation
– ... and perhaps reuse

Why SA?

Software Architecture as **Communication** !

- understanding, discussion, consensus...
- between all stakeholders

Software Architecture as **Design Plan**

- early analysis ⇒
  - risk assessment and risk management
  - optimise balance between different qualities
- bridge between requirements and implementation
  - implementation design decisions are determined by the architecture!

Why SA?

"Having an explicit and well-communicated architecture is the first step towards ensuring architectural conformance."

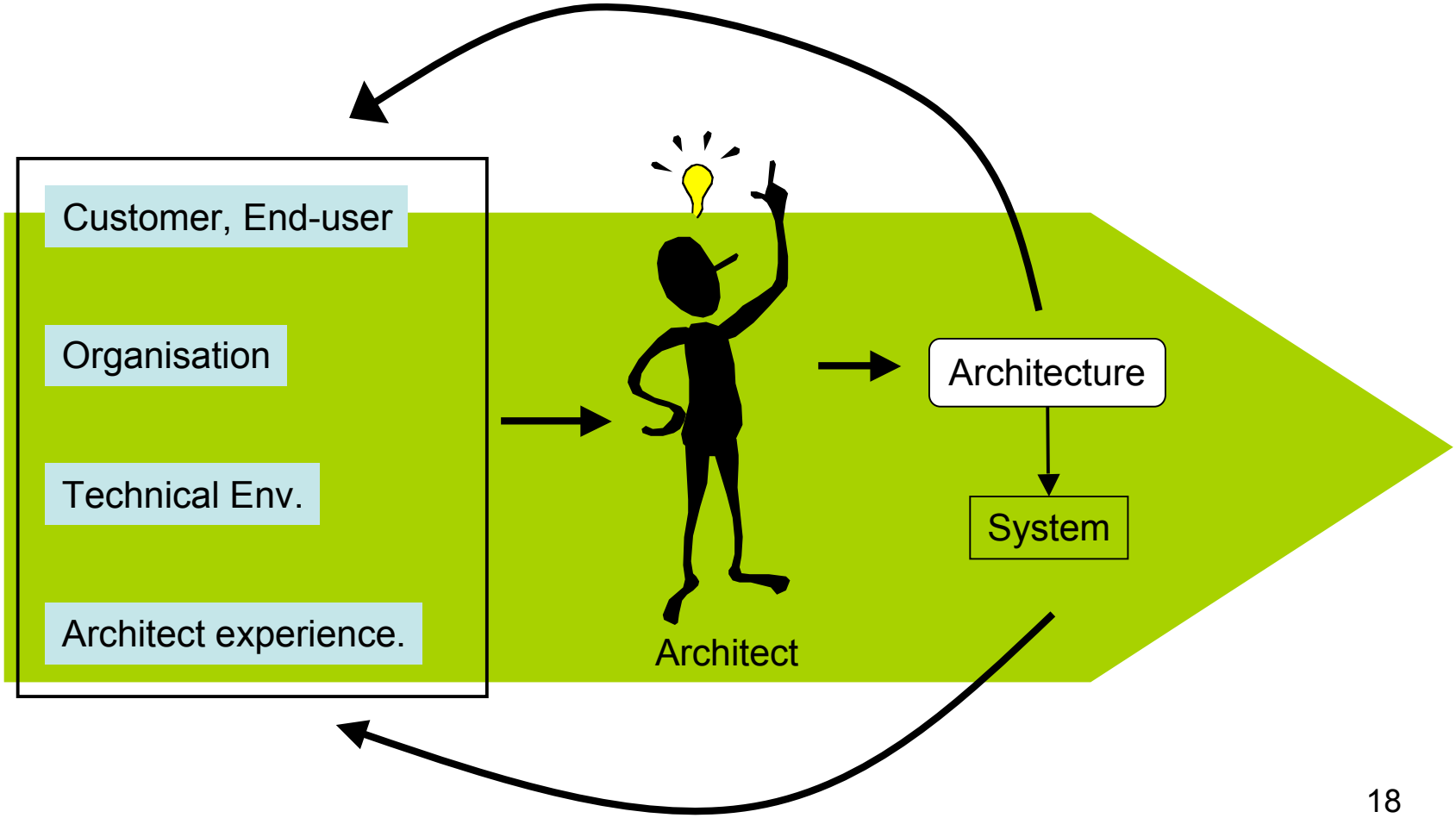
as-built corresponds to *as-designed*

AARHUS UNIVERSITET

Software Architecture and the Development Process

SA in a context

Architecture-Business-Cycle (ABC)



Software Architecture and the Development Process

When to software architect?

- How?
- Why?
- Who?

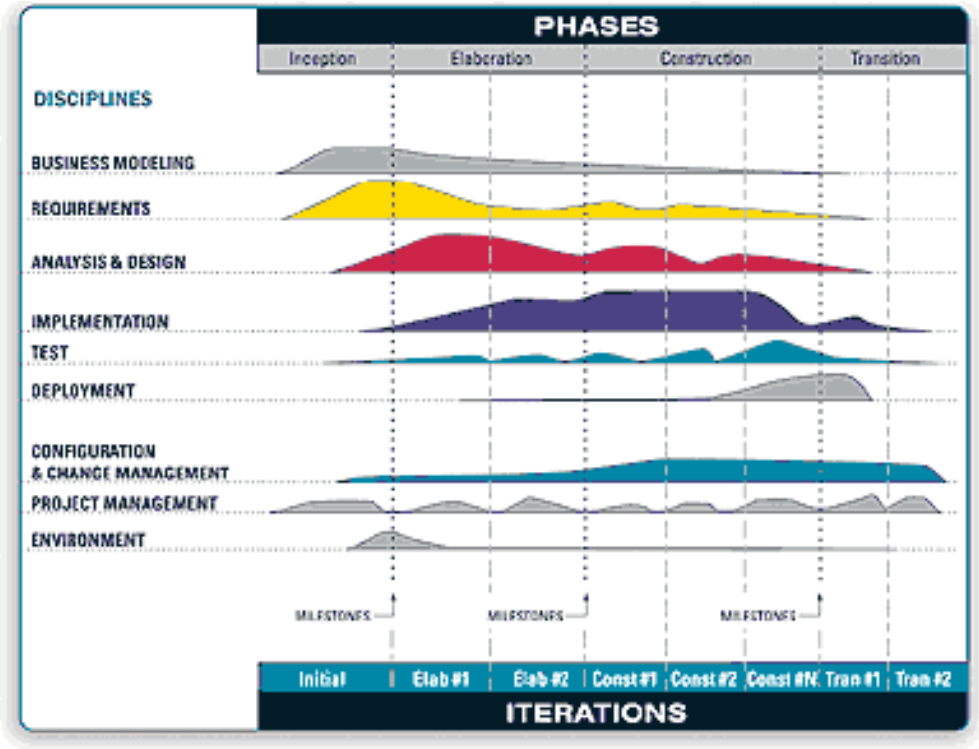
Three process examples

- (Rational) Unified Process
- eXtreme Programming
- Capability Maturity Model

Rational Unified Process

Use case-driven, “architecture-centric”, …

- Build skeletal system (architectural prototype)
- Fill in…



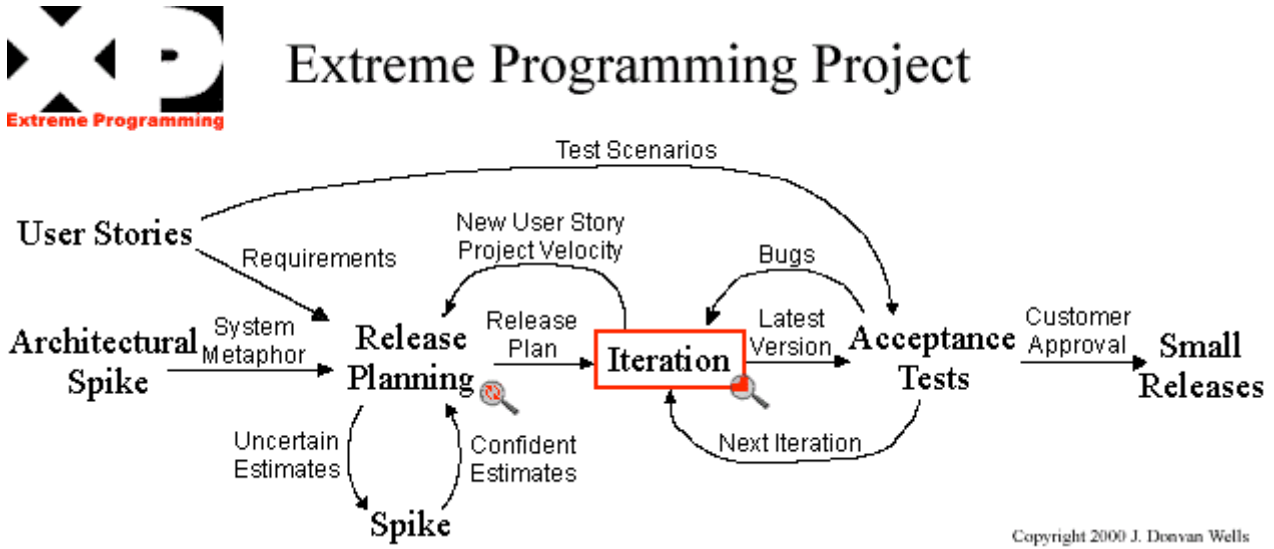
eXtreme Programming

Simplicity, communication, feedback, courage

- A set of practices: refactoring, unit testing, small releases

No software architecture per se in [Beck, 1999]

- "Metaphor" comes closest
- ("Architectural spike" below corresponds to an exploratory architectural prototype)



Capability Maturity Model

Level 1: Initial
- Ad hoc, person-dependent, blackbox process

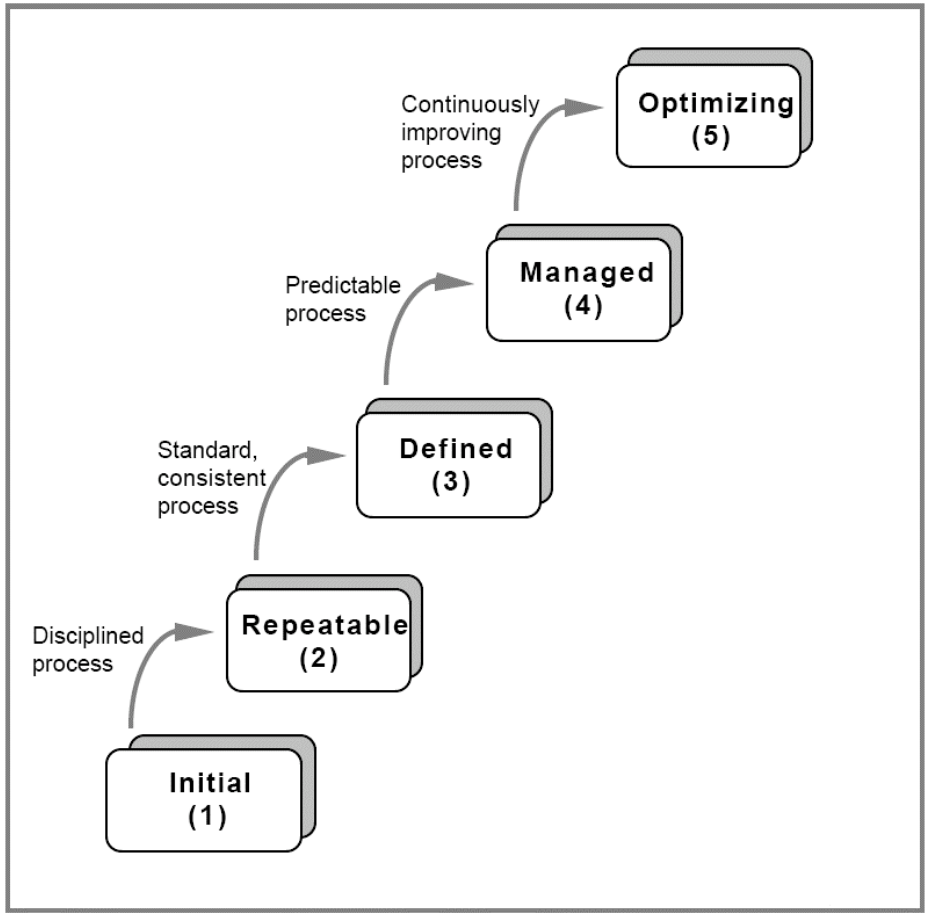
Level 2: Repeatable
- Basic management processes established
- Successes can be repeated

Level 3: Defined
- Documented and standardised processes for documentation and engineering

Level 4: Managed
- Detailed measurement of product and process quality

Level 5: Optimized
- Continuous process improvement
- Quantitative feedback and innovation

Software architecture part of Software Product Engineering
- Key process area for level 3

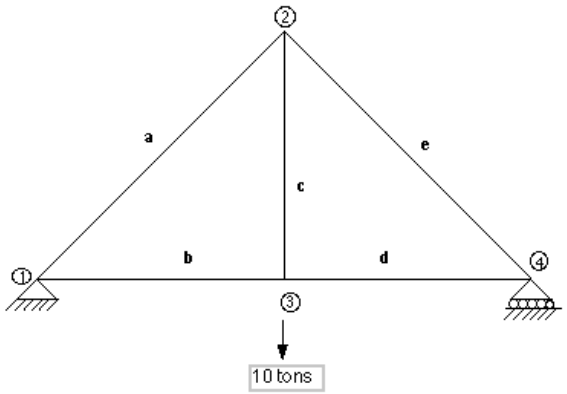


Views

Views

"A *view* represents a partial aspect of a software architecture that shows specific properties of a software system." [Buschmann]

Example: Consider a bridge

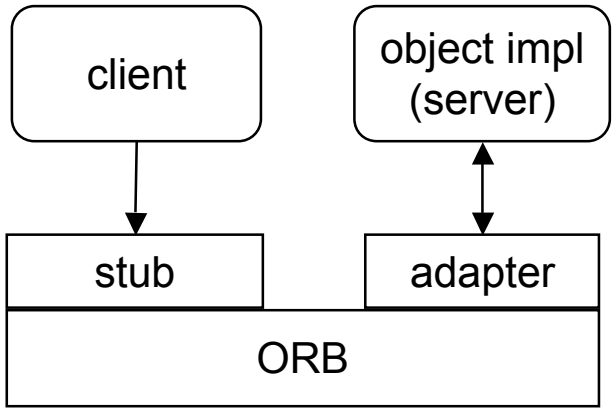


joint 2: Fx = fa(-cos45) + fc (cos90) + fe(cos45) = 0; Fx = - cfa + cfe = 0
joint 2: Fy = -fa(sin45) + - fc(sin90) + - fe(sin45) = 0; Fy = -sfa - fc - sfe = 0

Let's look at an example

Why not look at CORBA again…

Can 'views' explain CORBA better than this?

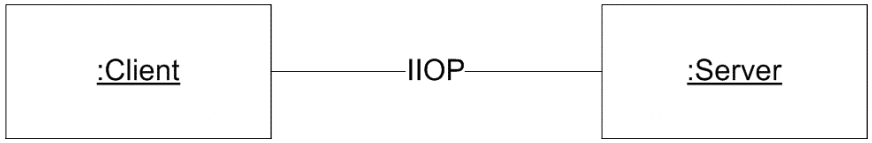


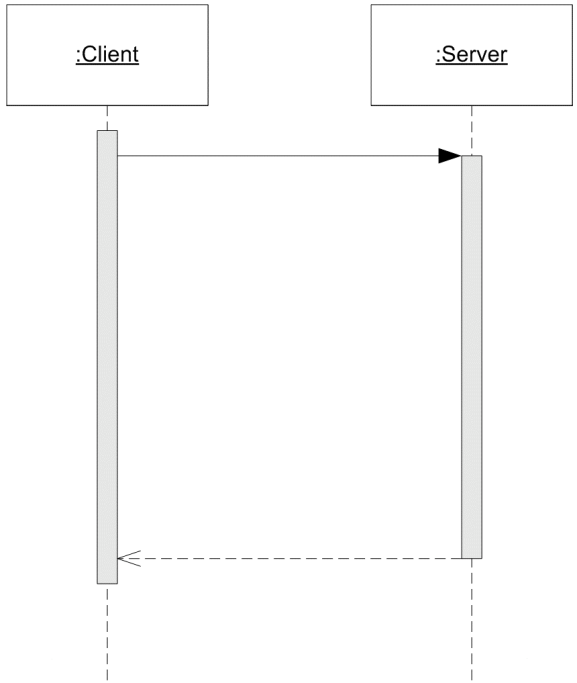
?

CORBA in Component Connector View (Dynamics)

CORBA is about *transparency*

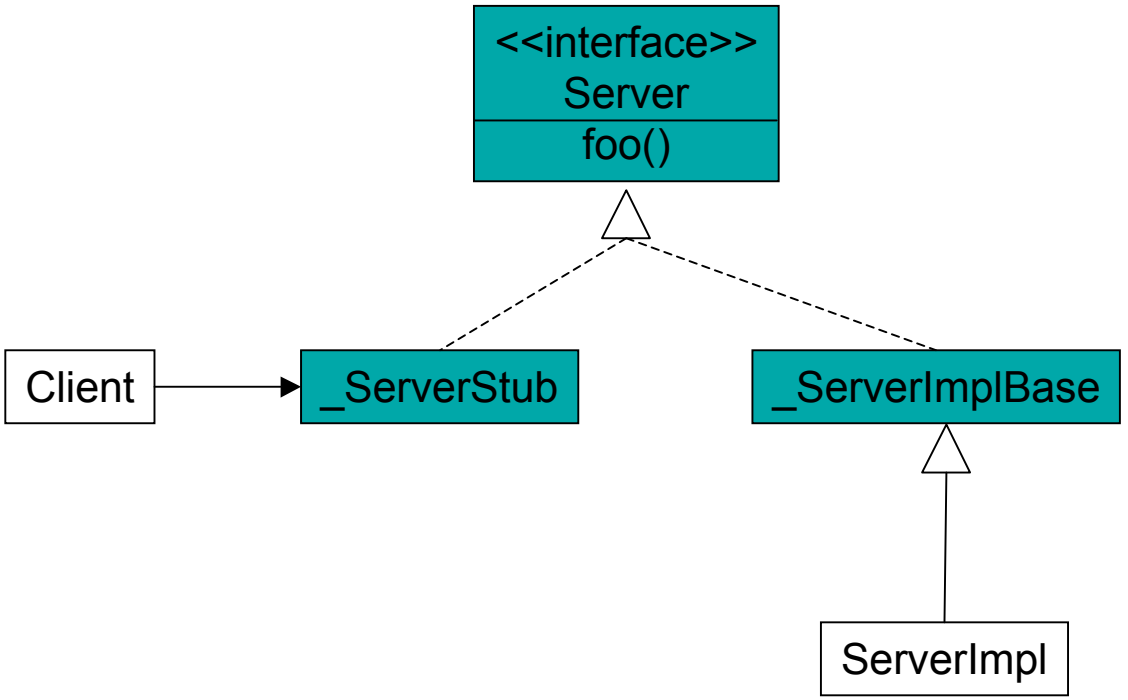
- abstract away the network
- programming model: *normal method call*





CORBA in Module View

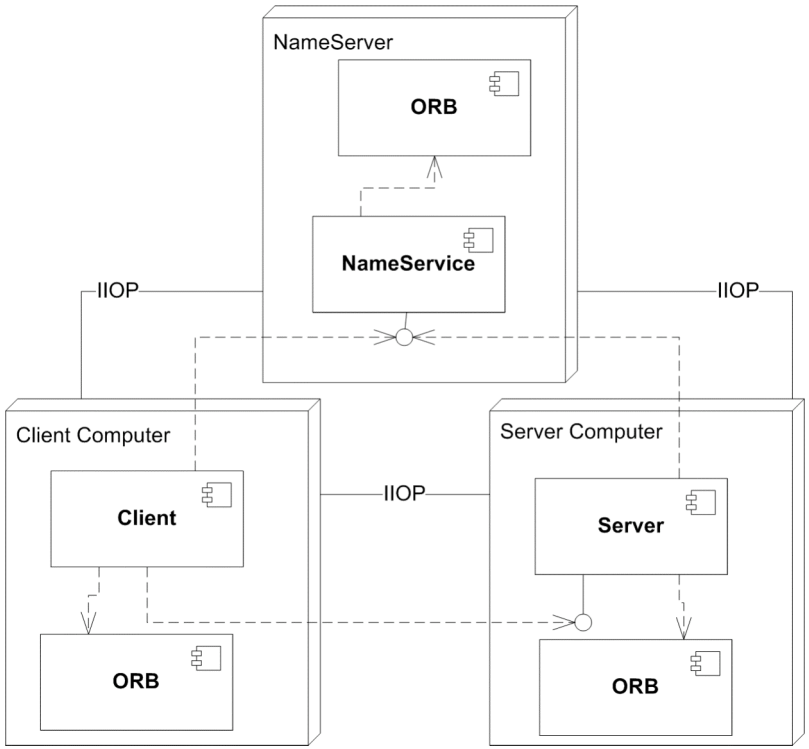
From the static/class view a *proxy* pattern is used



Blue classes are generated by IDL2Java compiler
White classes are written by you.

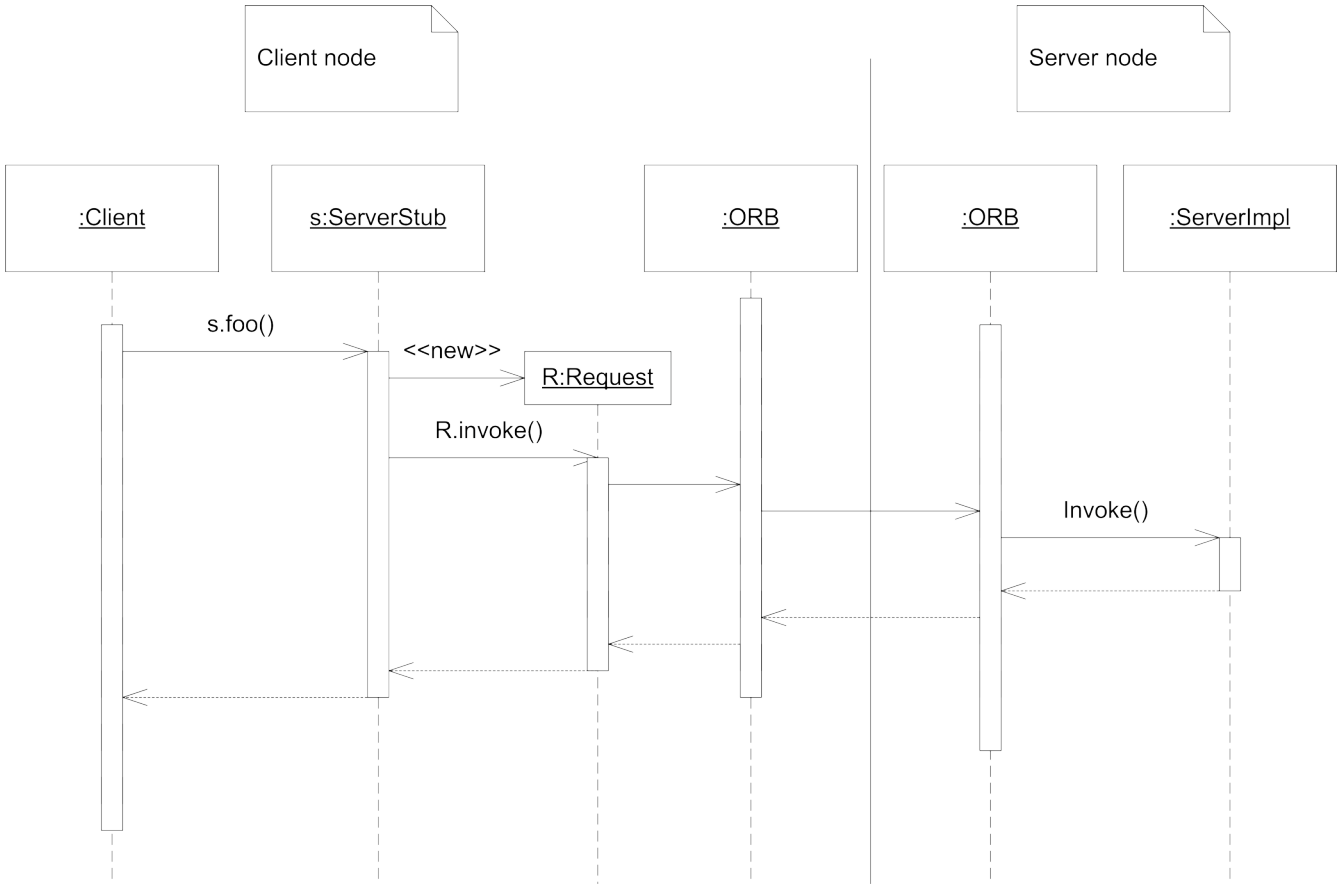
CORBA in Deployment View

Deployment-wise, the ORB middleware supports marshalling and unmarshalling ect.



CORBA Dynamics in more detail

The invocation dynamics is pretty complex

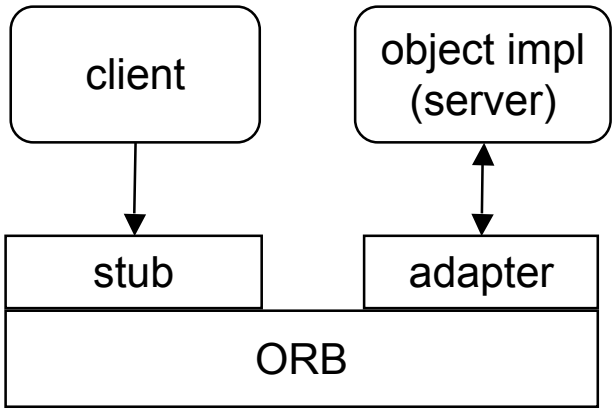


Morale

Views are important for precise communication and documentation!

I had to reverse engineer a Java CORBA example to really understand what was going on.

– This diagram is nonsense…



Discussion

Do you use ‘views’ in your documentation?

What views do you use?

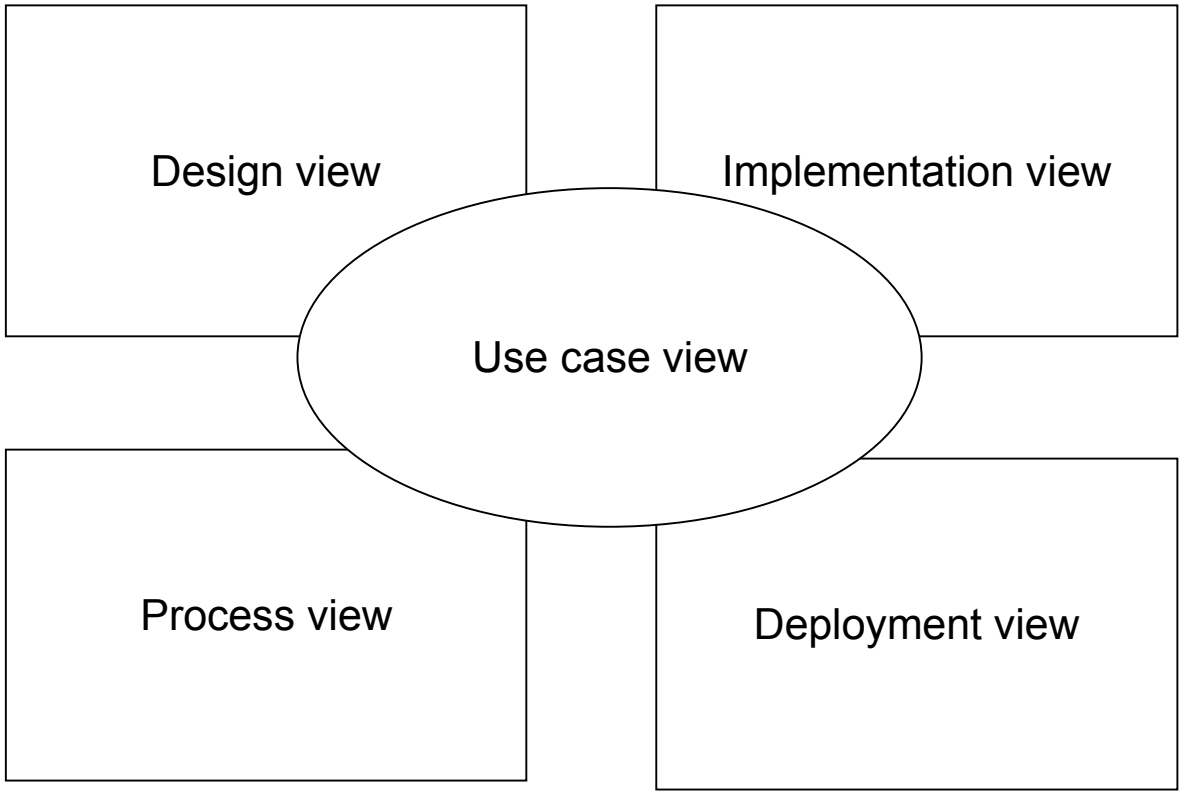
Are you strict about separating the deployment, class, dynamic, etc. structure?

The ”View Zoo”

Software architecture is a young research field.

Therefore different authors stress different views and what goes into them...

The Start: Kruchten 4+1 model



4+1 model [Rational]

Design View

- vocabulary of problem and its solution
- primarily supports functional requirements

Process view

- threads and processes / concurrency & synchronization
- addresses performance, scalability and throughput

4+1 model [Rational]

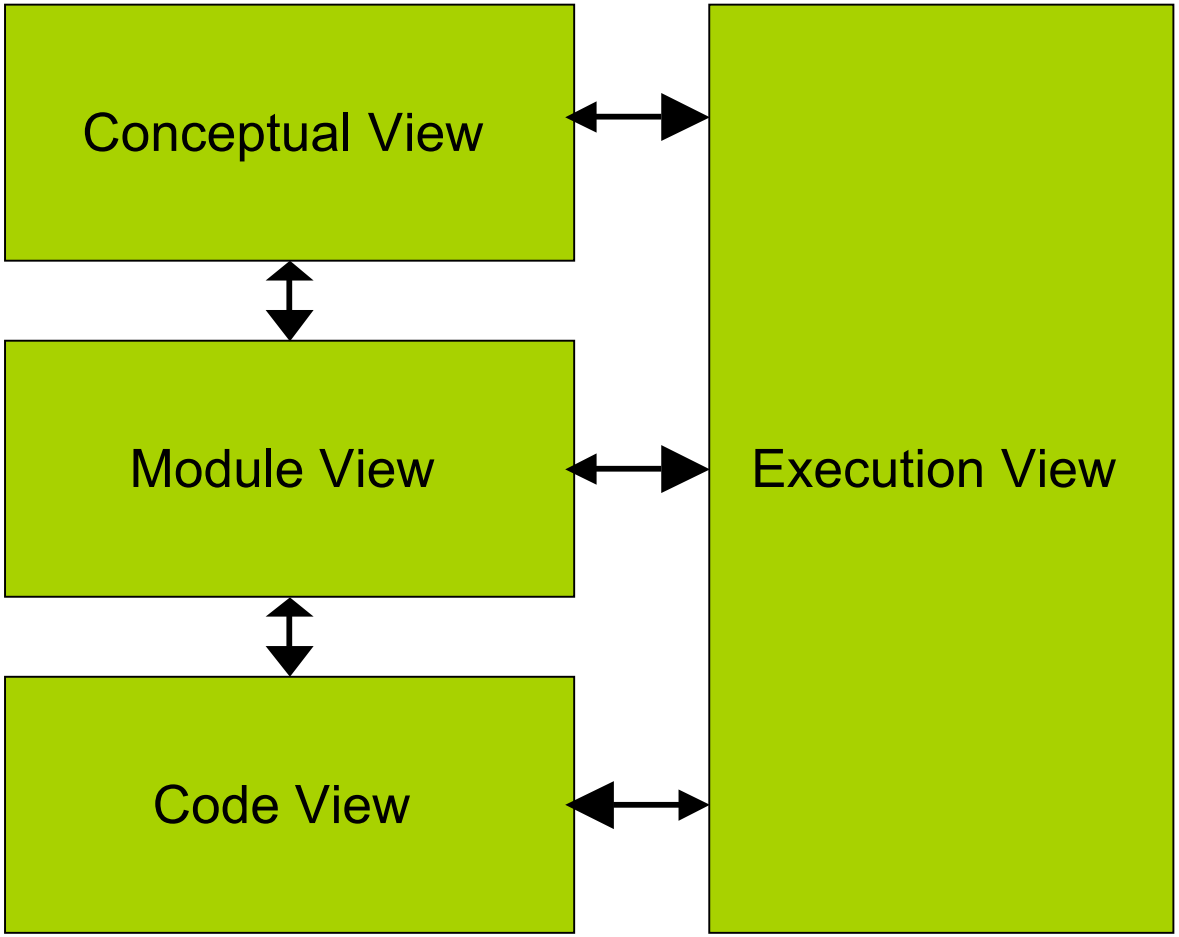
Implementation view

- components & files used to assemble and release the physical system
- addresses configuration management

Deployment view

- encompasses the nodes / hardware topology
- addresses distribution, delivery, installation of part that make up the physical system.

Hofmeister et al.



Hofmeister views

Logical/conceptual view

- deals with functionality

Module/implementation view

- deals with team structure, work-break-down, software configuration management and source control

Process/execution view

- deals with dynamics: performance, concurrency, synchronization

Deployment view

- deals with distribution, network performance, deployment at customer site

Enterprise Architecture (EA)

AARHUS UNIVERSITET

The architecture of an enterprise

- IT architecture and business architecture

Consider the Bass et al. definition of software architecture

> The software architecture of a computing system is the structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them [Bass et al., 2003]

- How does this relate to Enterprise Architecture?

Frameworks for EA

Zachman

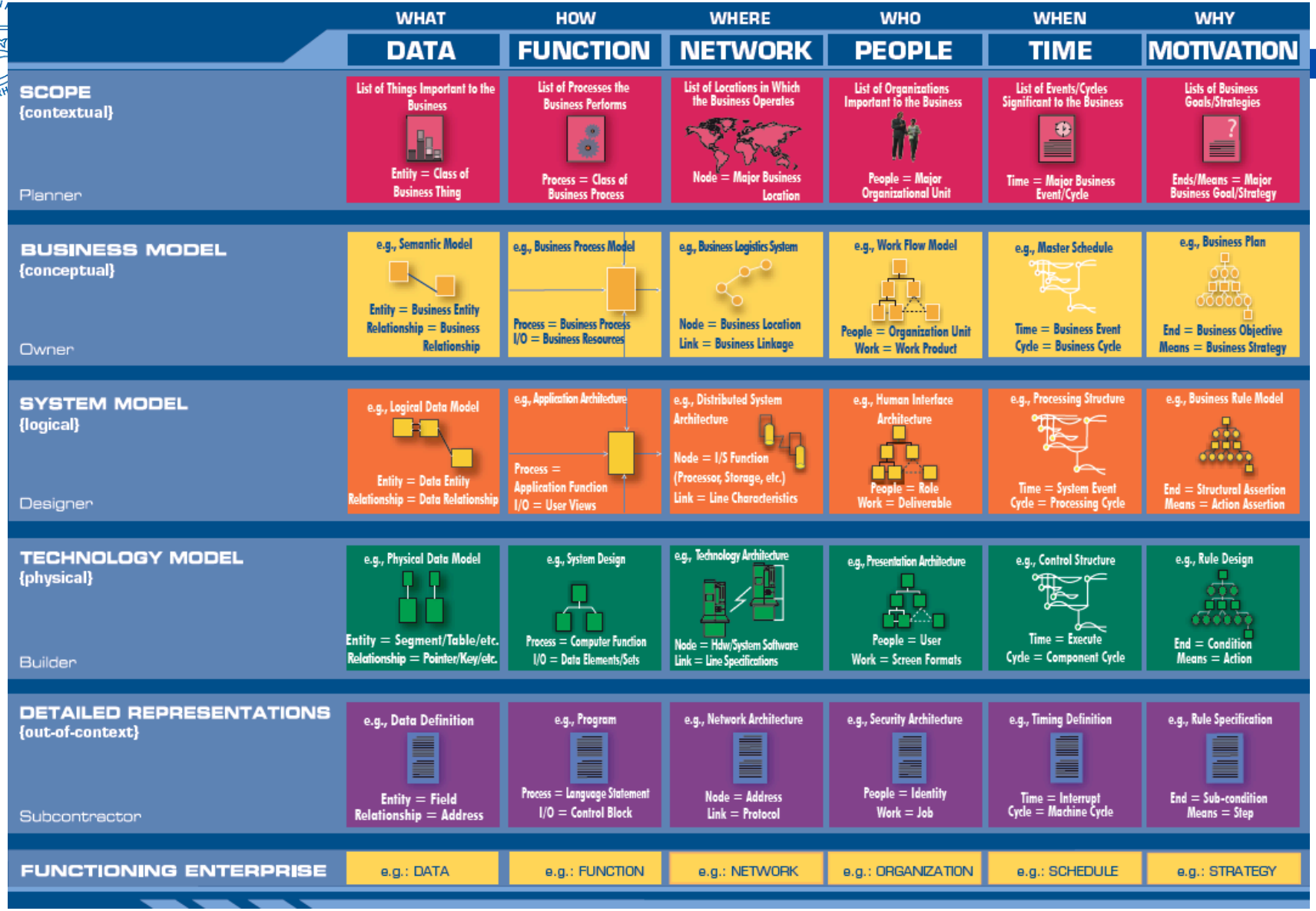
Department of Defense Architecture Framework (DODAF)

UK Ministry of Defence Architectural Framework (MODAF)

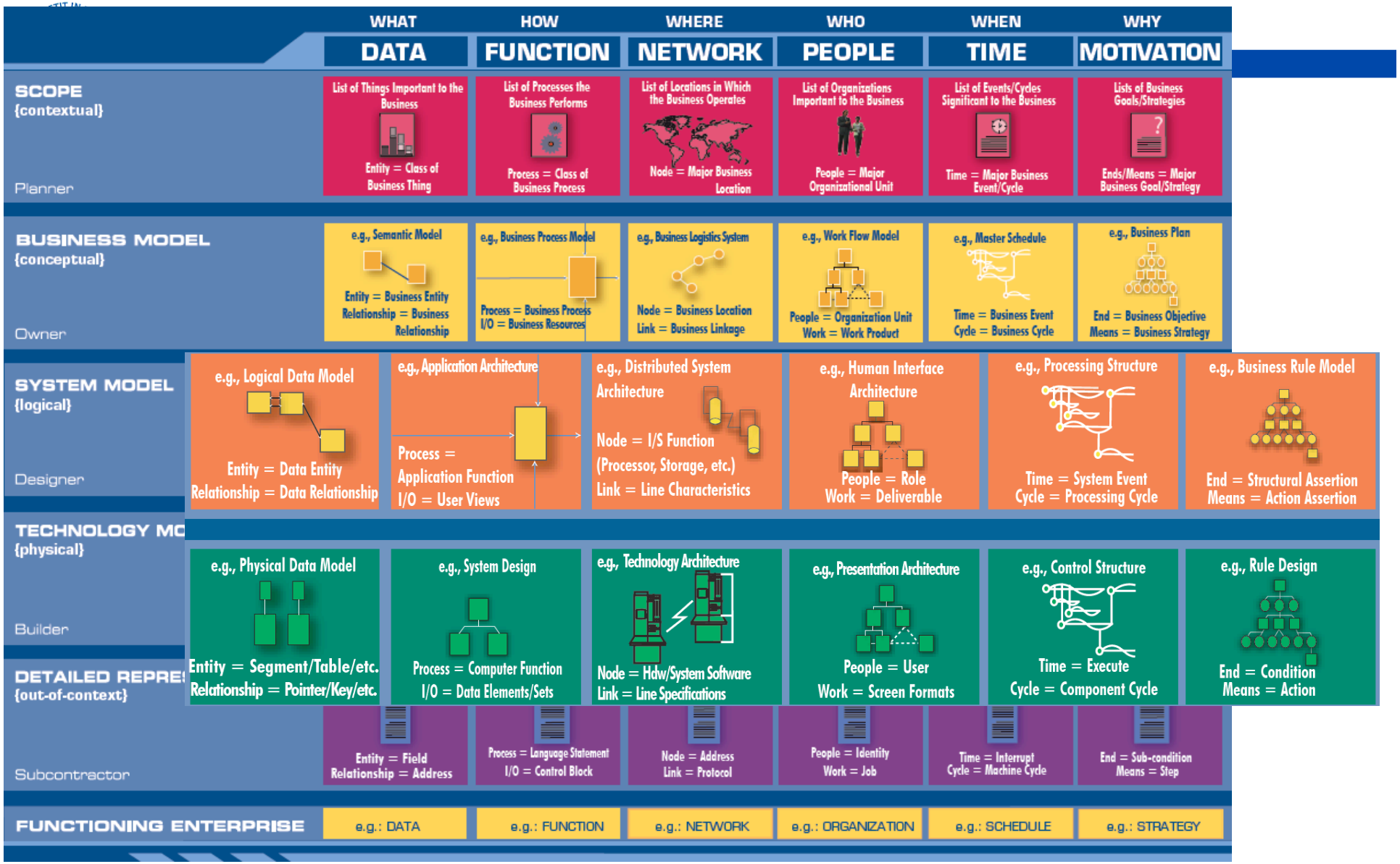
The Open Group Architecture Framework (TOGAF)

Offentlig Information Online (OIO) EA Framework

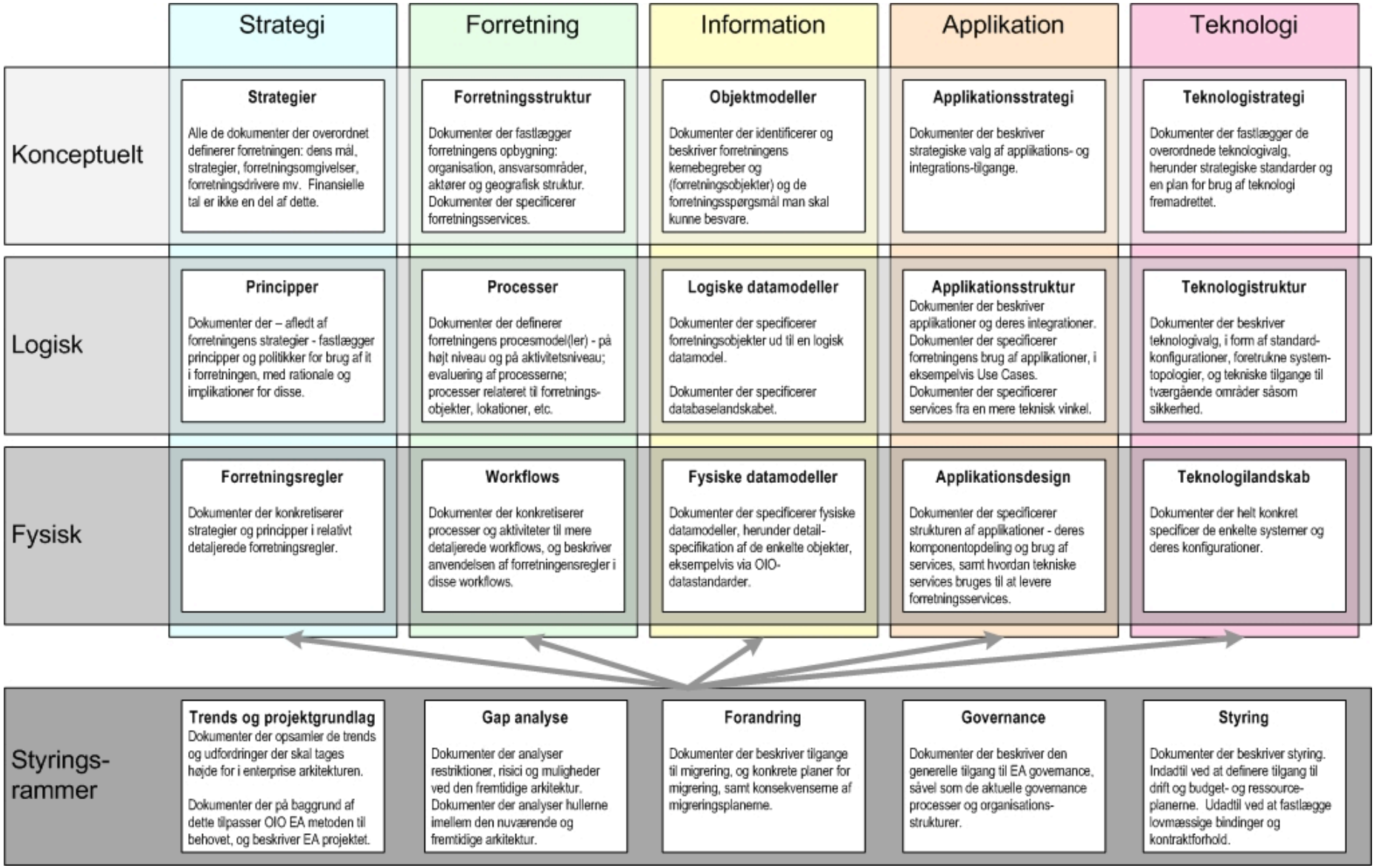
Zachman


WHAT
DATA
HOW
FUNCTION
WHERE
NETWORK
WHO
PEOPLE
WHEN
TIME
WHY
MOTIVATION
SCOPE
{contextual}
Planner
List of Things Important to the Business
Entity = Class of Business Thing
List of Processes the Business Performs
Process = Class of Business Process
List of Locations in Which the Business Operates
Node = Major Business Location
List of Organizations Important to the Business
People = Major Organizational Unit
List of Events/Cycles Significant to the Business
Time = Major Business Event/Cycle
Lists of Business Goals/Strategies
Ends/Means = Major Business Goal/Strategy
BUSINESS MODEL
{conceptual}
Owner
e.g., Semantic Model
Entity = Business Entity
Relationship = Business Relationship
e.g., Business Process Model
Process = Business Process
I/O = Business Resources
e.g., Business Logistics System
Node = Business Location
Link = Business Linkage
e.g., Work Flow Model
People = Organization Unit
Work = Work Product
e.g., Master Schedule
Time = Business Event
Cycle = Business Cycle
e.g., Business Plan
End = Business Objective
Means = Business Strategy
SYSTEM MODEL
{logical}
Designer
e.g., Logical Data Model
Entity = Data Entity
Relationship = Data Relationship
e.g., Application Architecture
Process = Application Function
I/O = User Views
e.g., Distributed System Architecture
Node = I/S Function (Processor, Storage, etc.)
Link = Line Characteristics
e.g., Human Interface Architecture
People = Role
Work = Deliverable
e.g., Processing Structure
Time = System Event
Cycle = Processing Cycle
e.g., Business Rule Model
End = Structural Assertion
Means = Action Assertion
TECHNOLOGY MODEL
{physical}
Builder
e.g., Physical Data Model
Entity = Segment/Table/etc.
Relationship = Pointer/Key/etc.
e.g., System Design
Process = Computer Function
I/O = Data Elements/Sets
e.g., Technology Architecture
Node = Hdw/System Software
Link = Line Specifications
e.g., Presentation Architecture
People = User
Work = Screen Formats
e.g., Control Structure
Time = Execute
Cycle = Component Cycle
e.g., Rule Design
End = Condition
Means = Action
DETAILED REPRESENTATIONS
{out-of-context}
Subcontractor
e.g., Data Definition
Entity = Field
Relationship = Address
e.g., Program
Process = Language Statement
I/O = Control Block
e.g., Network Architecture
Node = Address
Link = Protocol
e.g., Security Architecture
People = Identity
Work = Job
e.g., Timing Definition
Time = Interrupt
Cycle = Machine Cycle
e.g., Rule Specification
End = Sub-condition
Means = Step
FUNCTIONING ENTERPRISE
e.g.: DATA
e.g.: FUNCTION
e.g.: NETWORK
e.g.: ORGANIZATION
e.g.: SCHEDULE
e.g.: STRATEGY

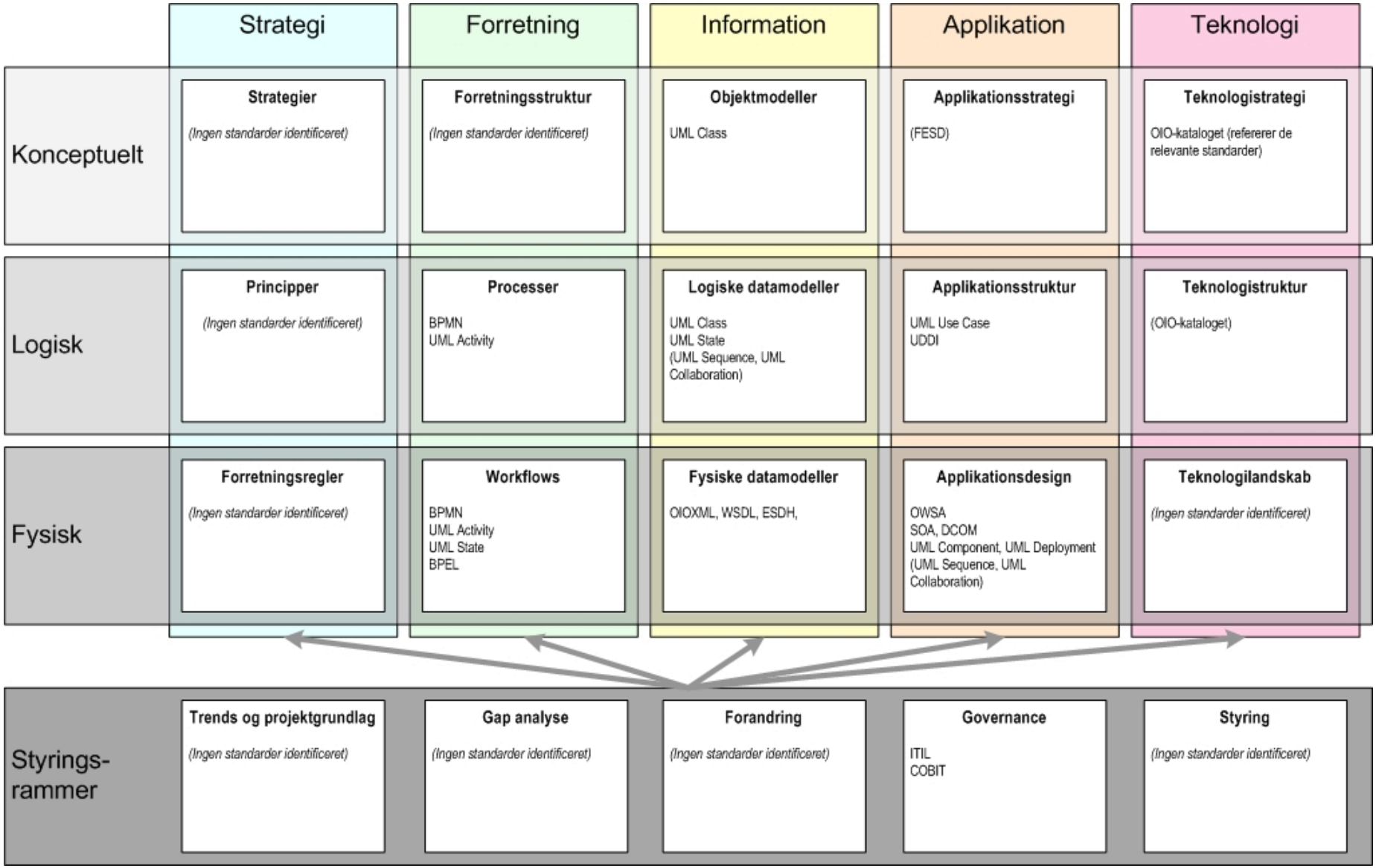
Zachman


WHAT
DATA
HOW
FUNCTION
WHERE
NETWORK
WHO
PEOPLE
WHEN
TIME
WHY
MOTIVATION
SCOPE
{contextual}
Planner
List of Things Important to the Business
Entity = Class of Business Thing
List of Processes the Business Performs
Process = Class of Business Process
List of Locations in Which the Business Operates
Node = Major Business Location
List of Organizations Important to the Business
People = Major Organizational Unit
List of Events/Cycles Significant to the Business
Time = Major Business Event/Cycle
Lists of Business Goals/Strategies
Ends/Means = Major Business Goal/Strategy
BUSINESS MODEL
{conceptual}
Owner
e.g., Semantic Model
Entity = Business Entity
Relationship = Business Relationship
e.g., Business Process Model
Process = Business Process
I/O = Business Resources
e.g., Business Logistics System
Node = Business Location
Link = Business Linkage
e.g., Work Flow Model
People = Organization Unit
Work = Work Product
e.g., Master Schedule
Time = Business Event
Cycle = Business Cycle
e.g., Business Plan
End = Business Objective
Means = Business Strategy
SYSTEM MODEL
{logical}
Designer
e.g., Logical Data Model
Entity = Data Entity
Relationship = Data Relationship
e.g., Application Architecture
Process = Application Function
I/O = User Views
e.g., Distributed System Architecture
Node = I/S Function (Processor, Storage, etc.)
Link = Line Characteristics
e.g., Human Interface Architecture
People = Role
Work = Deliverable
e.g., Processing Structure
Time = System Event
Cycle = Processing Cycle
e.g., Business Rule Model
End = Structural Assertion
Means = Action Assertion
TECHNOLOGY MO
{physical}
Builder
e.g., Physical Data Model
Entity = Segment/Table/etc.
Relationship = Pointer/Key/etc.
e.g., System Design
Process = Computer Function
I/O = Data Elements/Sets
e.g., Technology Architecture
Node = Hdw/System Software
Link = Line Specifications
e.g., Presentation Architecture
People = User
Work = Screen Formats
e.g., Control Structure
Time = Execute
Cycle = Component Cycle
e.g., Rule Design
End = Condition
Means = Action
DETAILED REPRE
{out-of-context}
Subcontractor
Entity = Field
Relationship = Address
Process = Language Statement
I/O = Control Block
Node = Address
Link = Protocol
People = Identity
Work = Job
Time = Interrupt
Cycle = Machine Cycle
End = Sub-condition
Means = Step
FUNCTIONING ENTERPRISE
e.g.: DATA
e.g.: FUNCTION
e.g.: NETWORK
e.g.: ORGANIZATION
e.g.: SCHEDULE
e.g.: STRATEGY

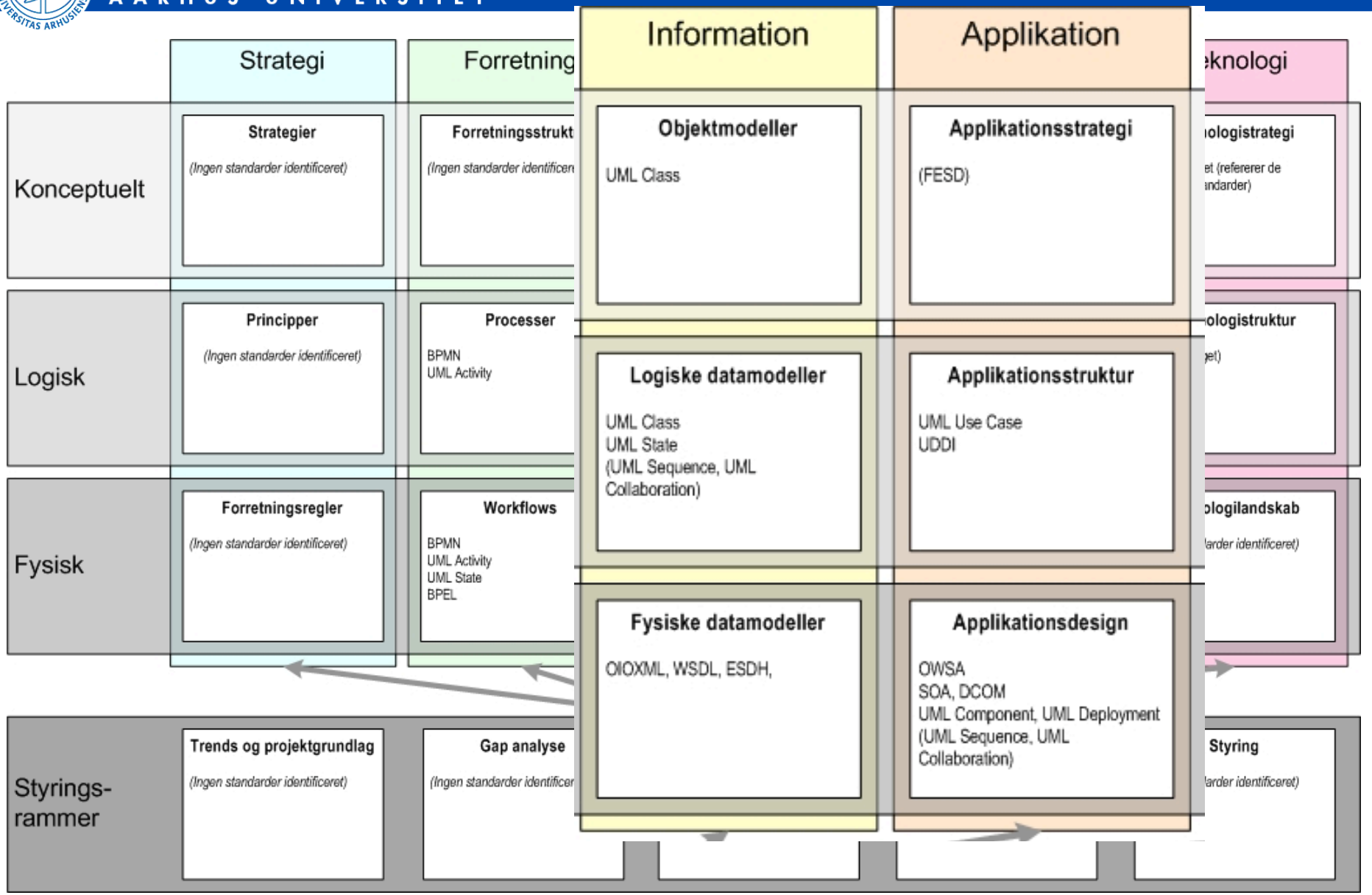
OIO EA Framework



OIO EA Framework



OIO EA Framework



Views

Brooks in his ”No Silver Bullet”-paper states:

Software is invisible and unvisualizable... In spite of progress in restricting and simplifying the structures of software, they remain inherently unvisualizable, thus depriving the mind of some of its most powerful conceptual tools.

Benefits of views

The most important benefit:

The recognition that there is no single view on a software system!

Clear, precise, focus in diagrams, documentation and discussions.

Benefits of views

Handling complexity

- **separation of concern:** the 'big mudpile' is partitioned into distinct views = aspects of the architecture
- compact notation with precise (?) meaning

Form checklist over aspects that you ought to consider relevant

Documentation is decoupled ⇒ more robust to changes...

Our Suggestion

We suggest a *minimalist* approach*:*

- C&C view: Focus on run-time behavior
- Module view: Focus on compile-time elements
- Deployment view: Focus on computing nodes

  - Inspired by Clements et al. 2002

Your experience

Are you using any of these views?

Or others?

What are the most appropriate for your domain?

Discussion

But...

- what views are important?
- what aspects are part of what views?
- what to call the views?
- what notation to use?
- can we use UML to describe architecture?

The central point is

- important to consider the relevance of central views
- important to split the documentation
- important to have compact notation that central (all?) stakeholders understand