Formal methods Research at NTNU Department of Telematics (ITEM)

FMICS 20003 Røros

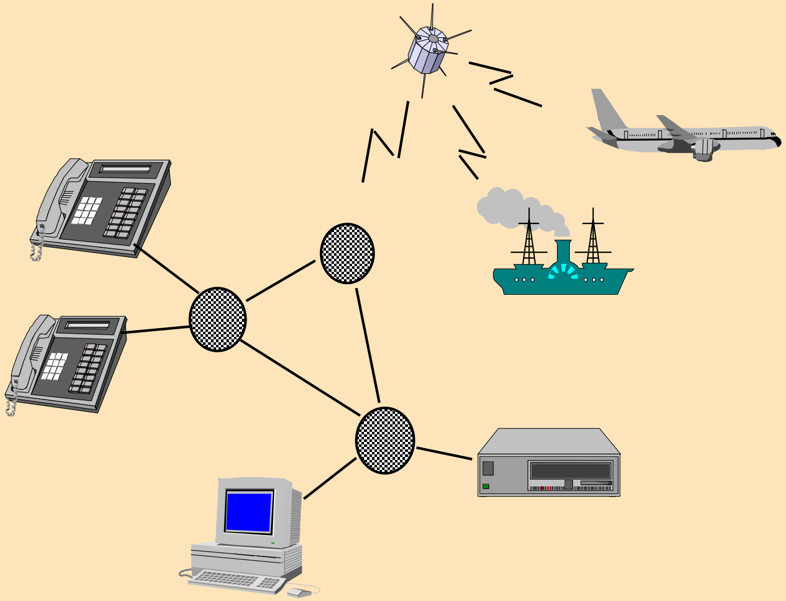
By

Rolv Bræk

Trondheim, June 2003

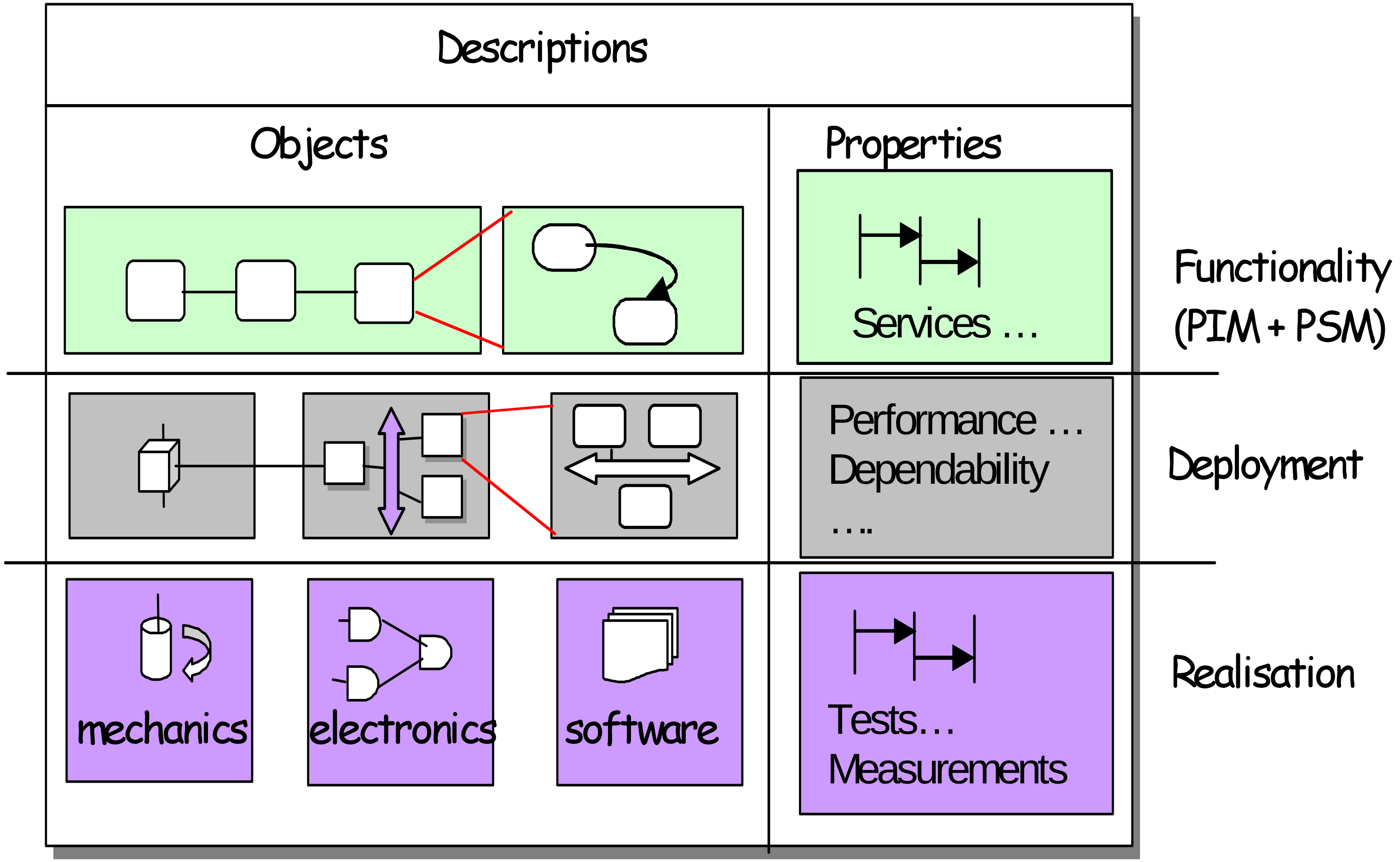
Our focus is on:

- The real-time and telecom domain:
  - Distributed systems
  - Highly parallel and
  - time dependent behaviour
  - High dependability
  - High performance



- Stateful reactive behaviour
- Industrial applications
- Constructive methods (before corrective)
- Rapid service engineering, dynamic adaptation and deployment

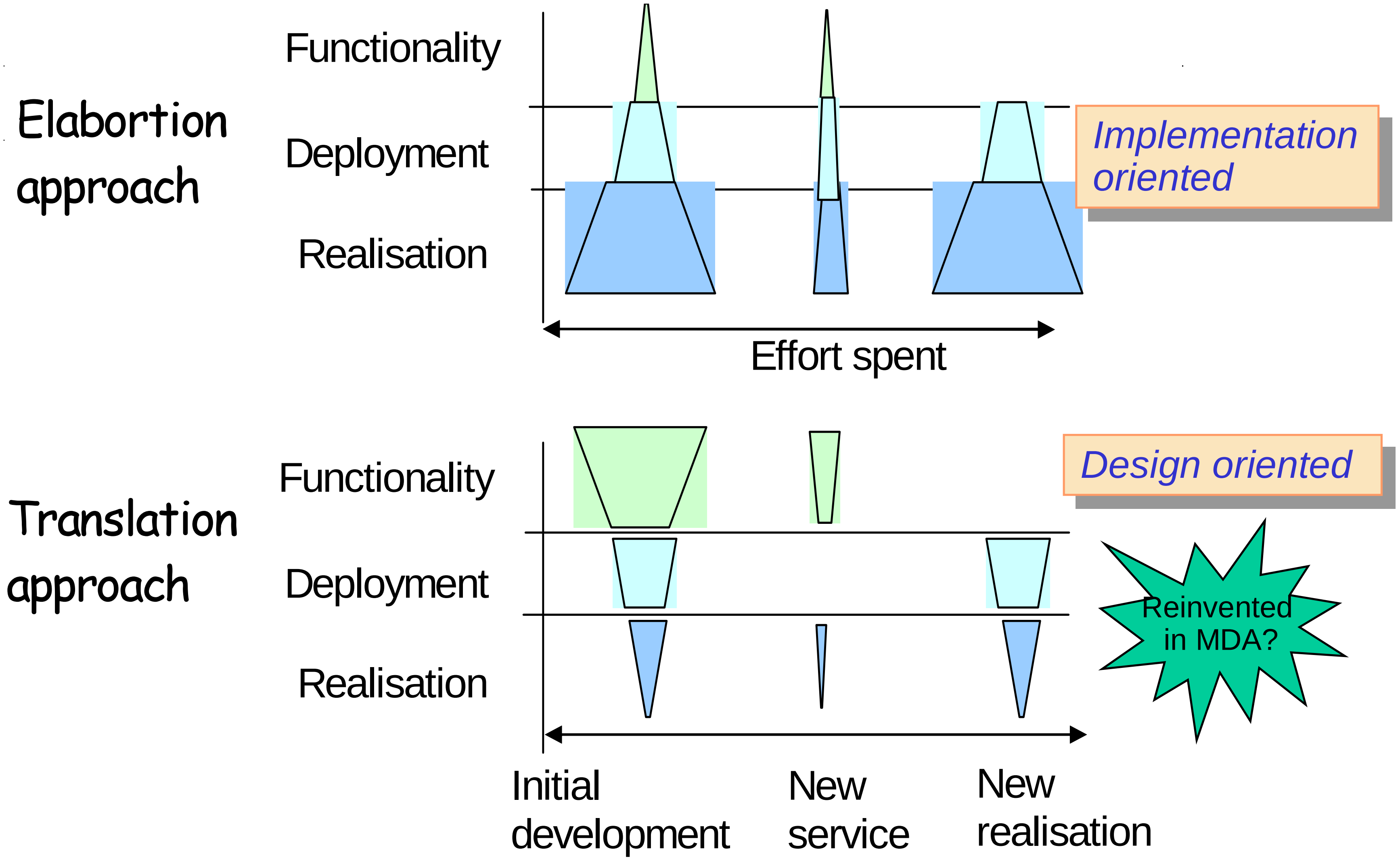
Our core model architecture



…using the ITU-T languages and UML

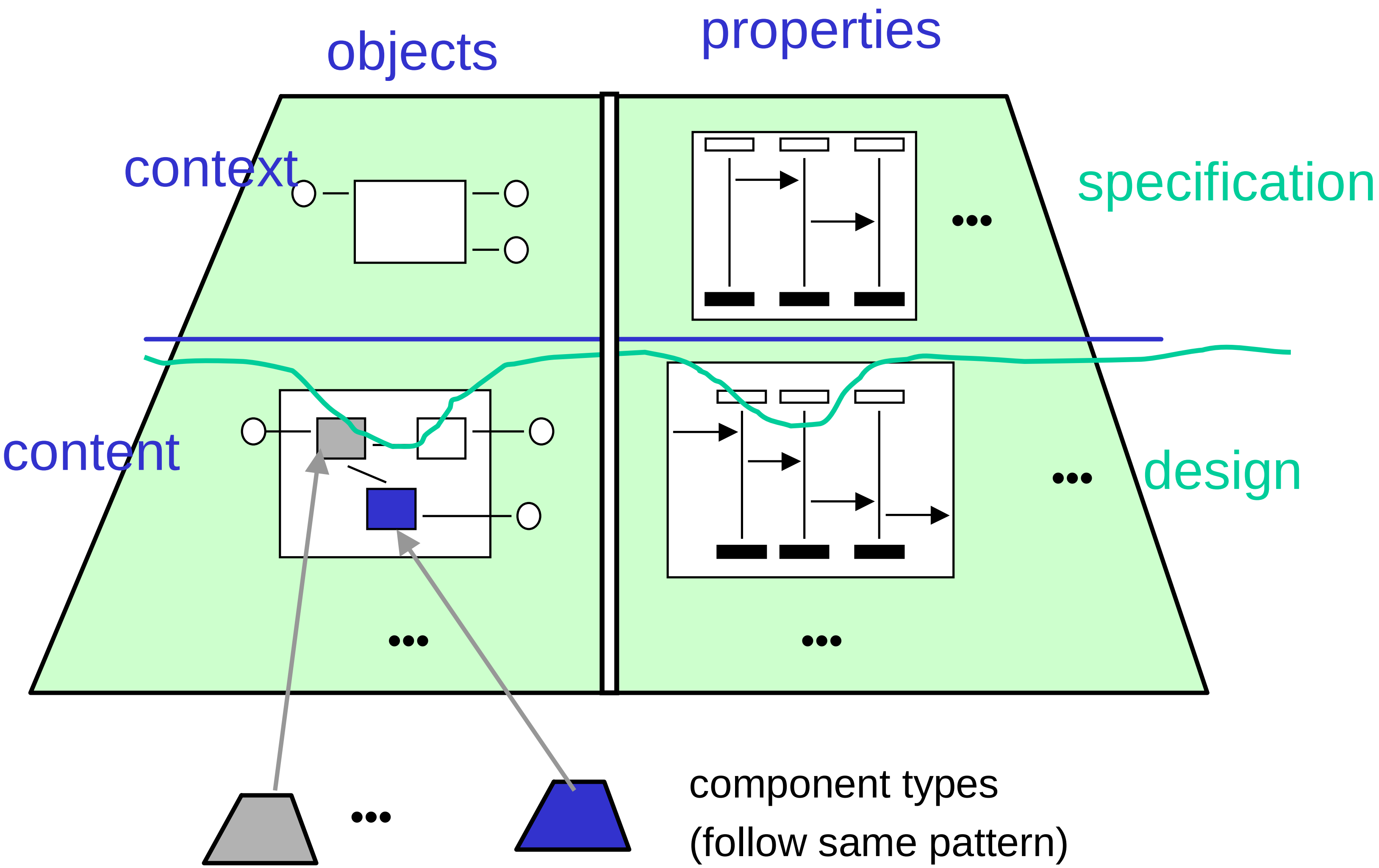
ITU-T		UML		
Objects	Properties	Objects	Properties	
UMLsdl, SDL, ASN.1	MSC,	Class, State-Machines	UseCase, Sequence, Collaboration, OCL, Activity	Functionality
	TTCN, MSC	Deployment, Component, Class	Sequence, Collaboration, OCL	Deployment
CHILL, ASN.1	TTCN, MSC		Sequence, Collaboration, OCL	Realisation

… and the translation approach



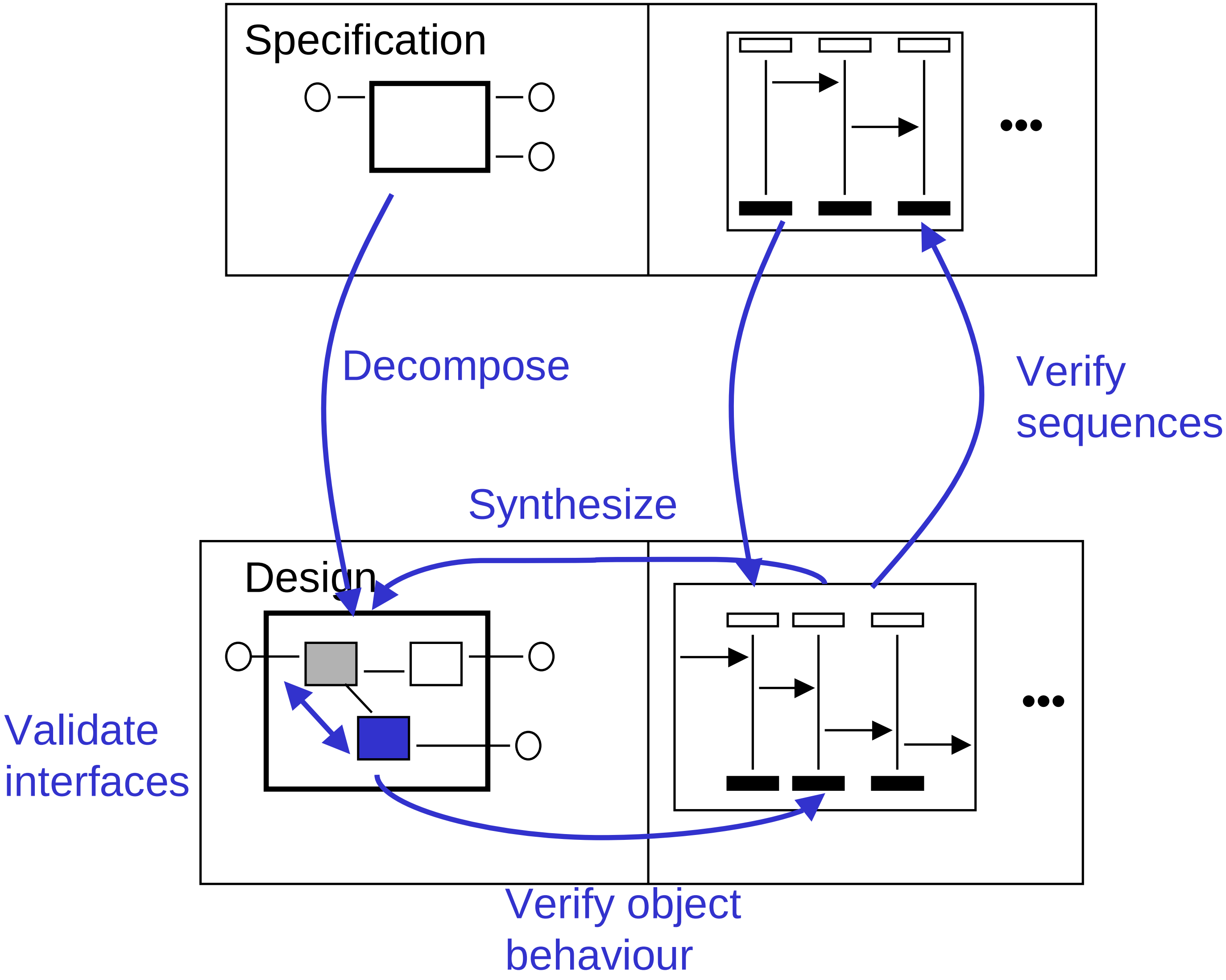
Trondheim, June 2003

… with evolving models



component types
(follow same pattern)

... being aligned and consistent



Specification

Design

Decompose

Synthesize

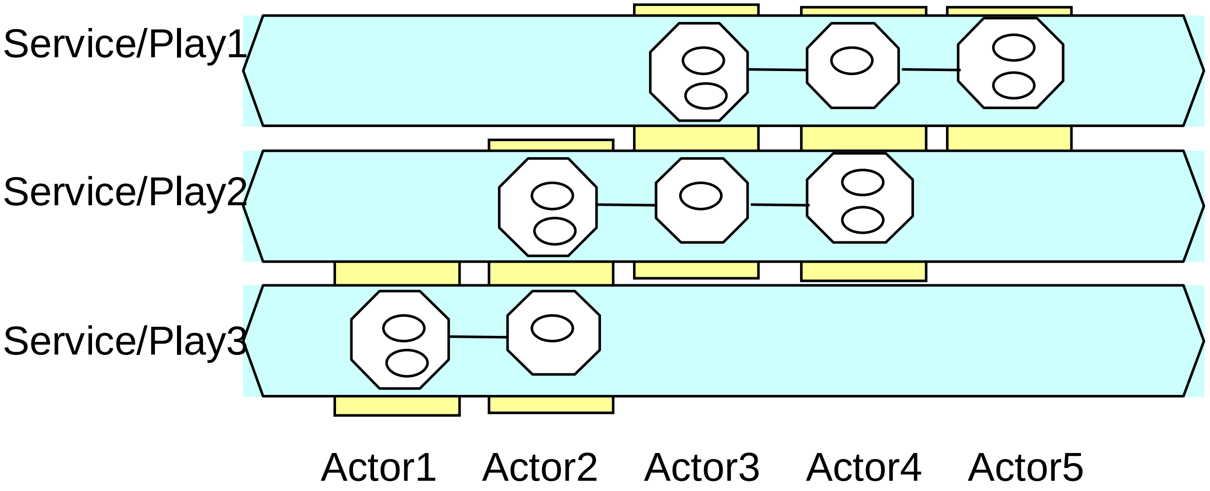
Verify sequences

Validate interfaces

Verify object behaviour

Towards service modularity

- Services are dynamic structure of of co-operating actors playing different roles, i.e. a kind of play.
- Actors take part in several plays and play several roles simultaneously.
- Service modularity require that roles be modelled and designed separately and then composed dynamically in a manner that enable desirable plays and avoids undesirable plays.



A service role is:

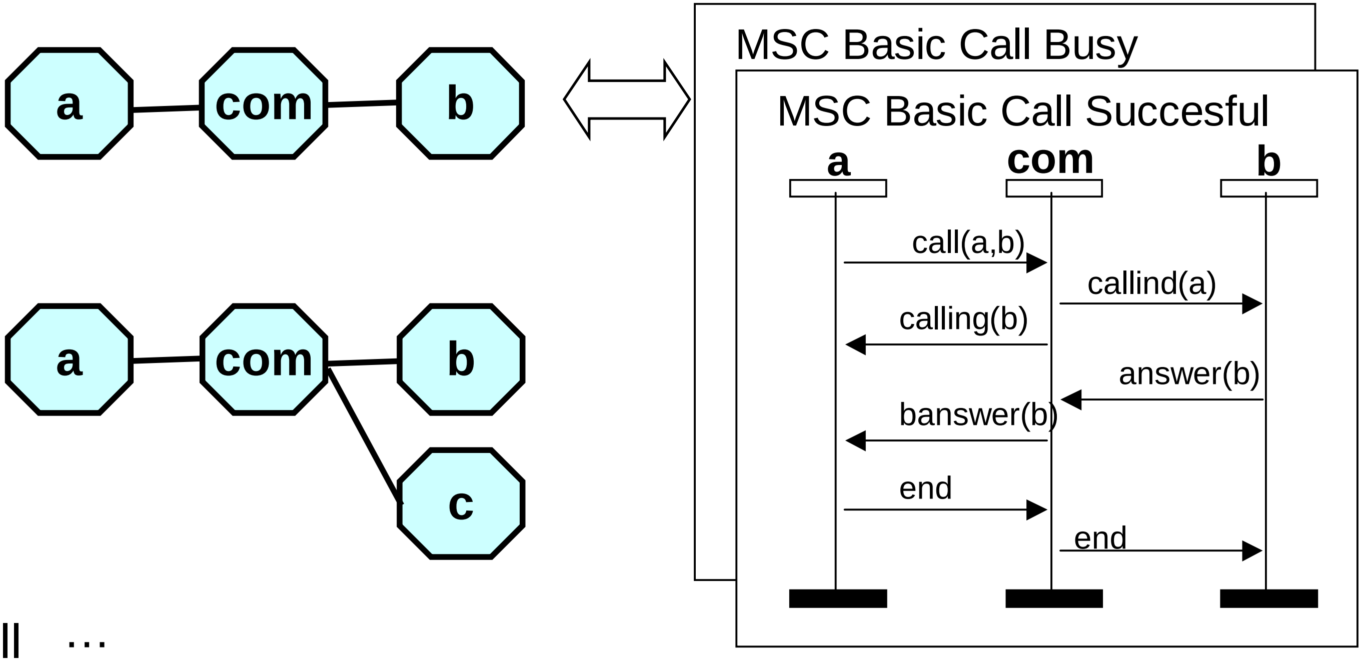
- the part some object plays in a service
- used to model services separately

Services — Service roles: s-roles — Service role behaviour

Basic Call

Enquiry Call

Conference Call ...



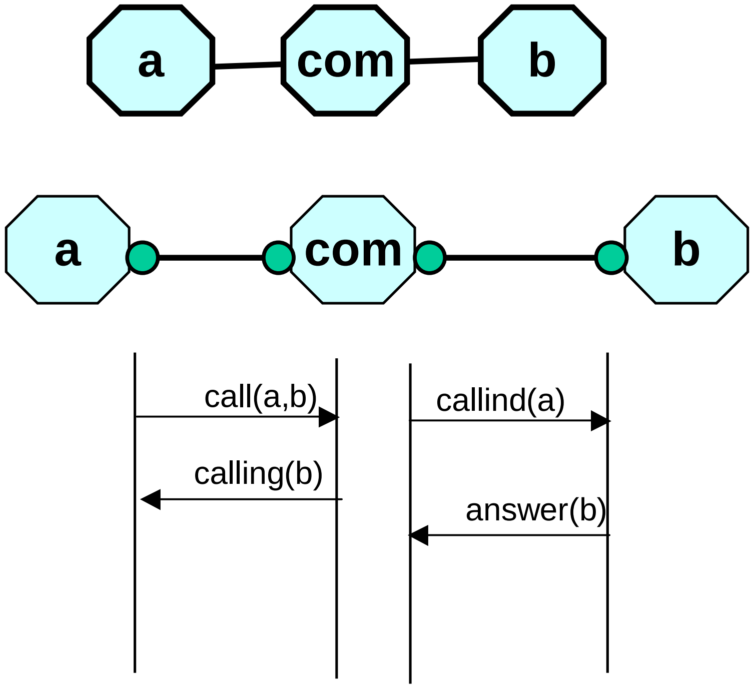
An association role is:

- the part of a behaviour visible on an association end or interface
- used to define interface behaviours

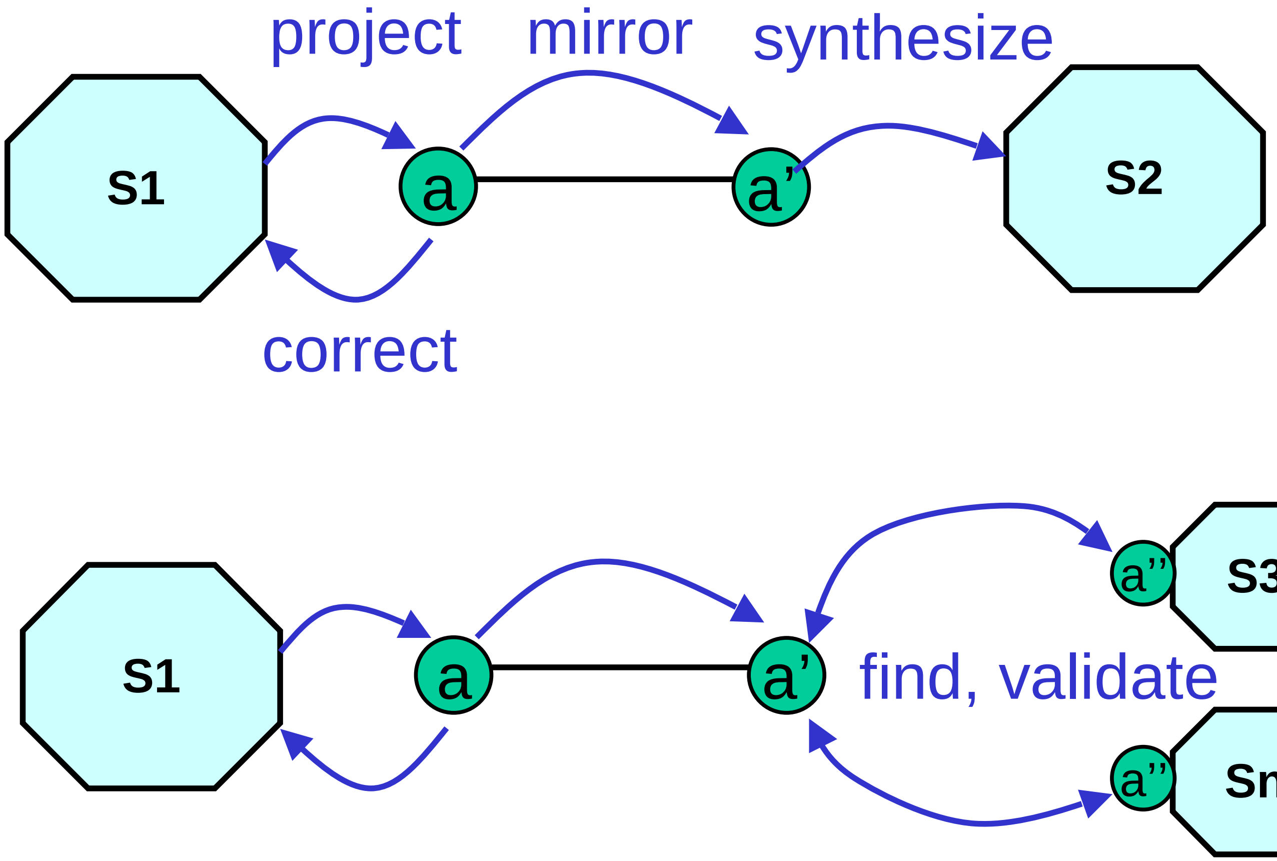
Service roles

Association roles:
a-roles

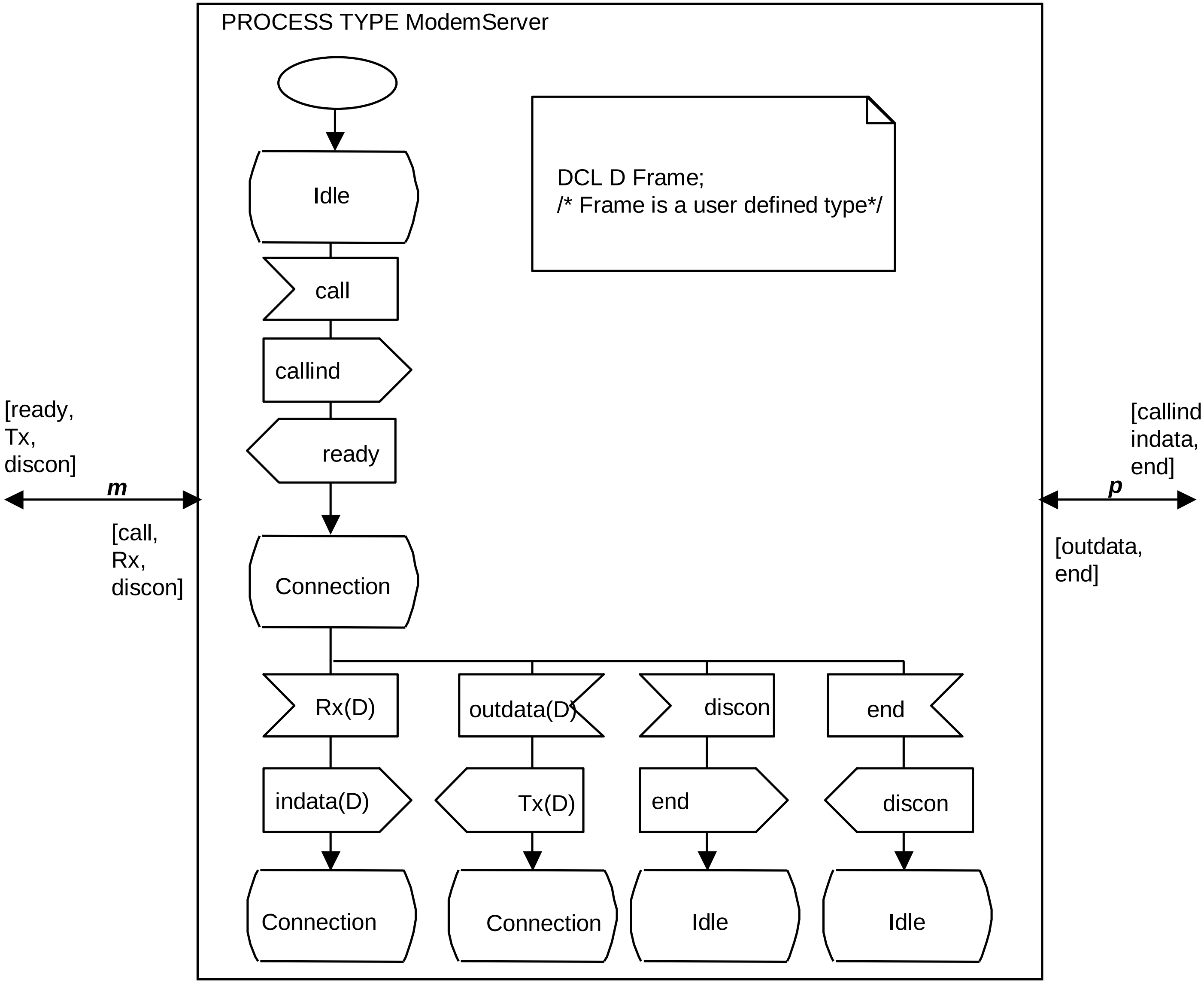
a-role behaviour



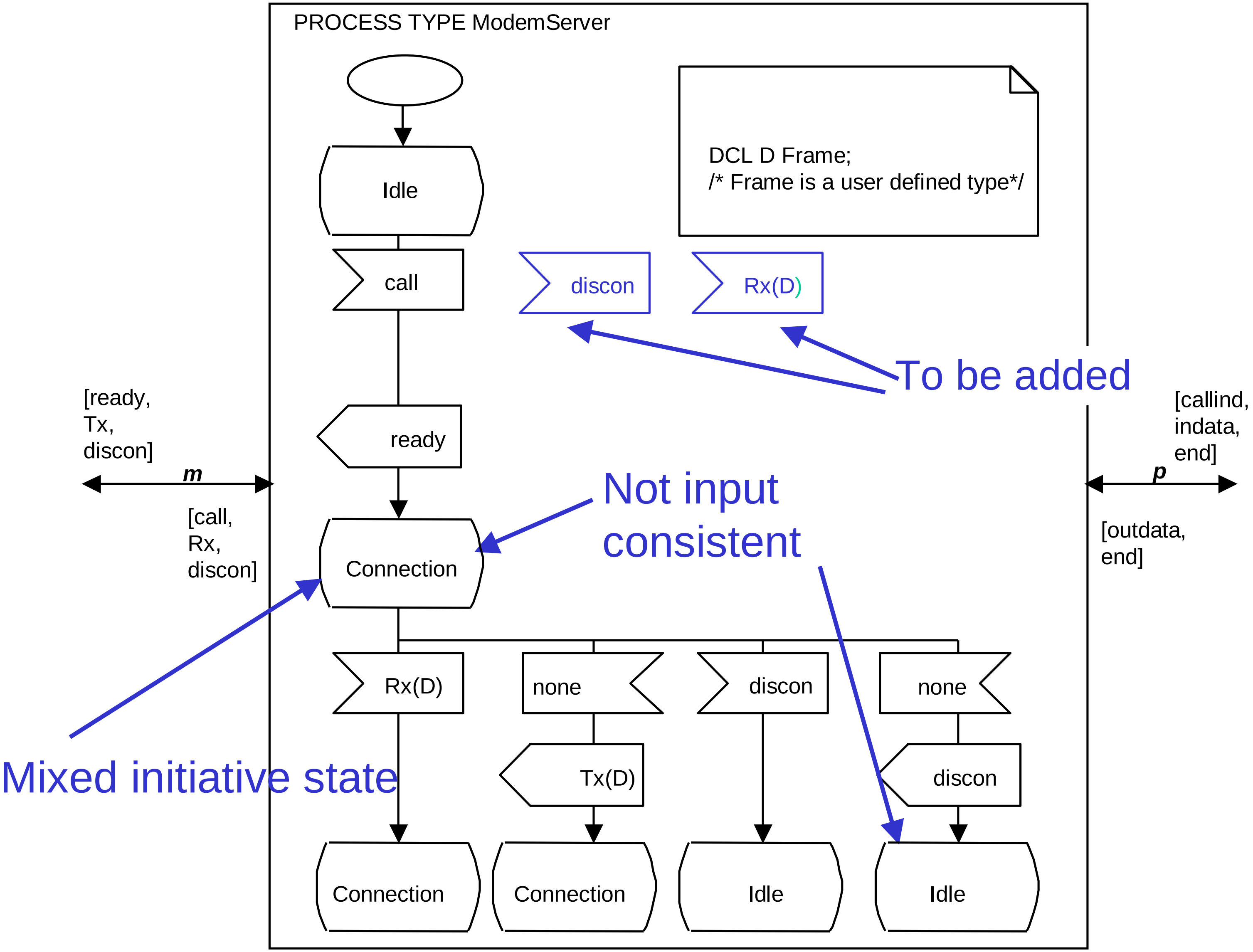
Working with a_roles



Are there any problems here?



a-role towards m: two problems revealed

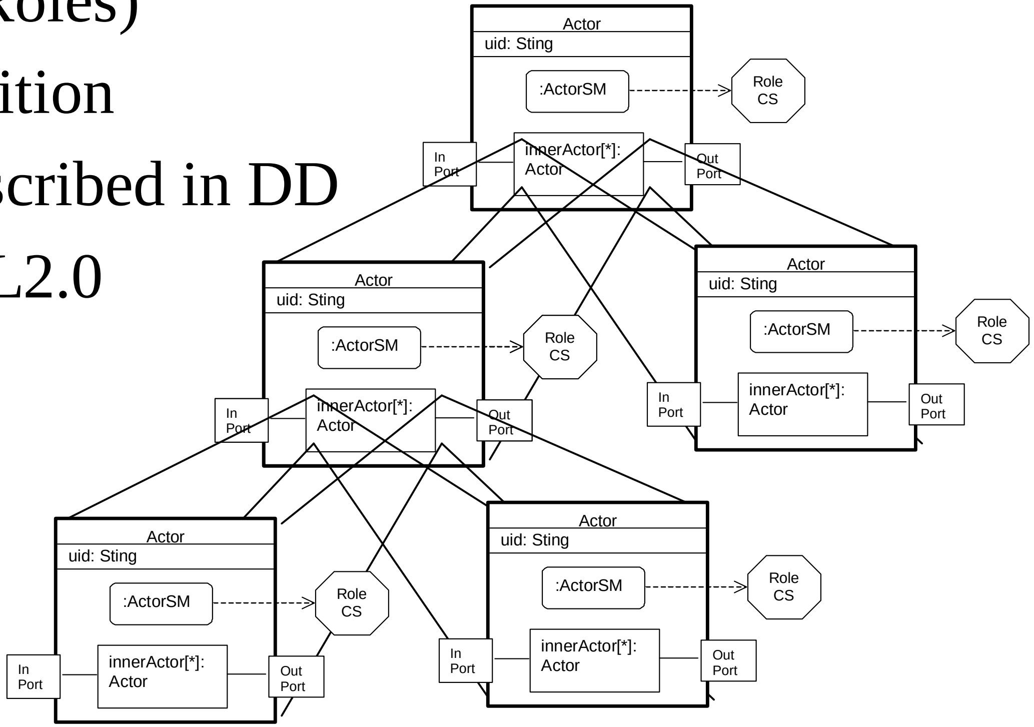


Roles are like projections and useful for:

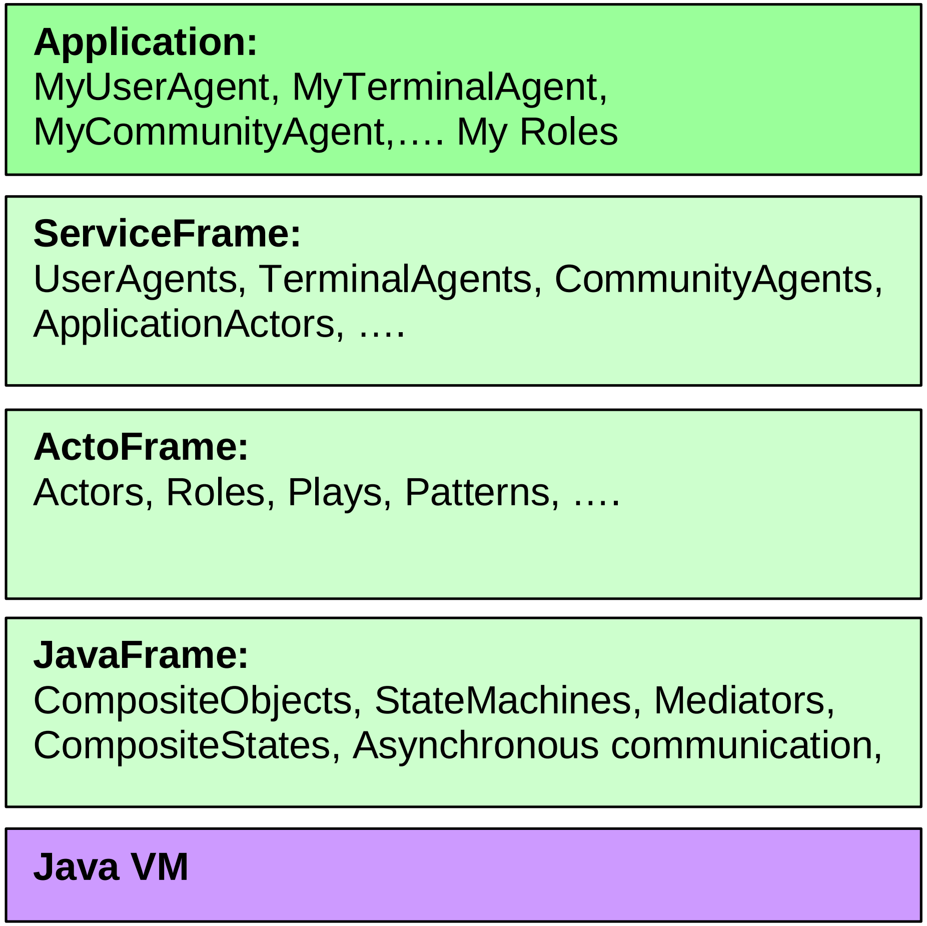
- Architecture definitions: a-roles help to define interfaces precisely
- Design verification: comparing required roles with the design
- Link validation: provided a-roles must “contain” required a-roles
- Finding design flaws: projections must be consistent
- Reuse: roles define interfaces and reusable functionality
- Design: roles serve as specifications and role designs as (reusable) composition units

Implementing using ActorFrame

- Roles are played by Actors
- An actor controls the lifecycle for its inner Actors(Roles)
- Hierarchical composition
- Internal structure described in DD
- Compliant with UML2.0



Implementation Layering



Application:
MyUserAgent, MyTerminalAgent,
MyCommunityAgent,…. My Roles

ServiceFrame:
UserAgents, TerminalAgents, CommunityAgents,
ApplicationActors, ….

Provides Application domain concepts

ActoFrame:
Actors, Roles, Plays, Patterns, ….

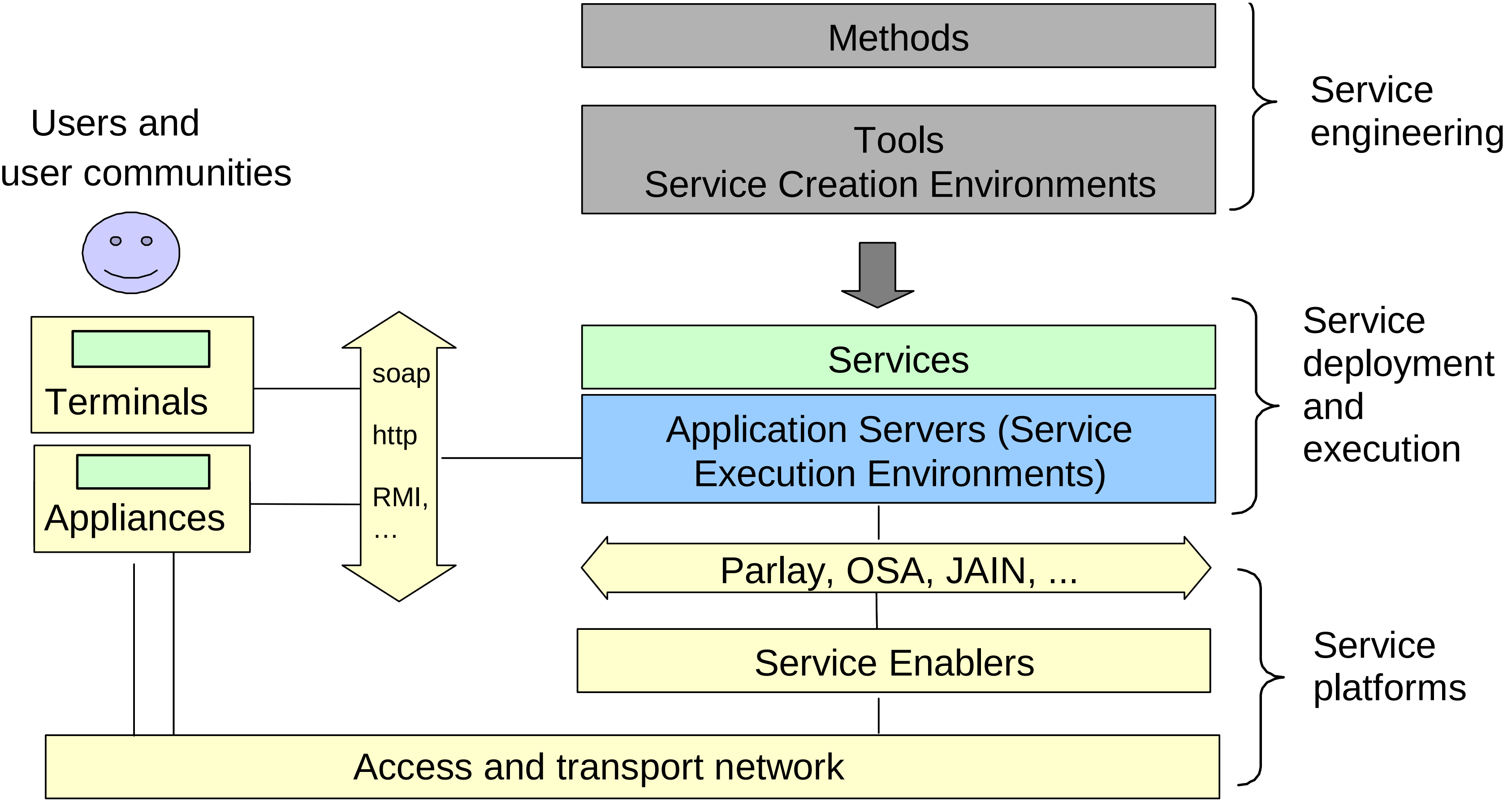
Provides Role modeling concepts

JavaFrame:
CompositeObjects, StateMachines, Mediators,
CompositeStates, Asynchronous communication,

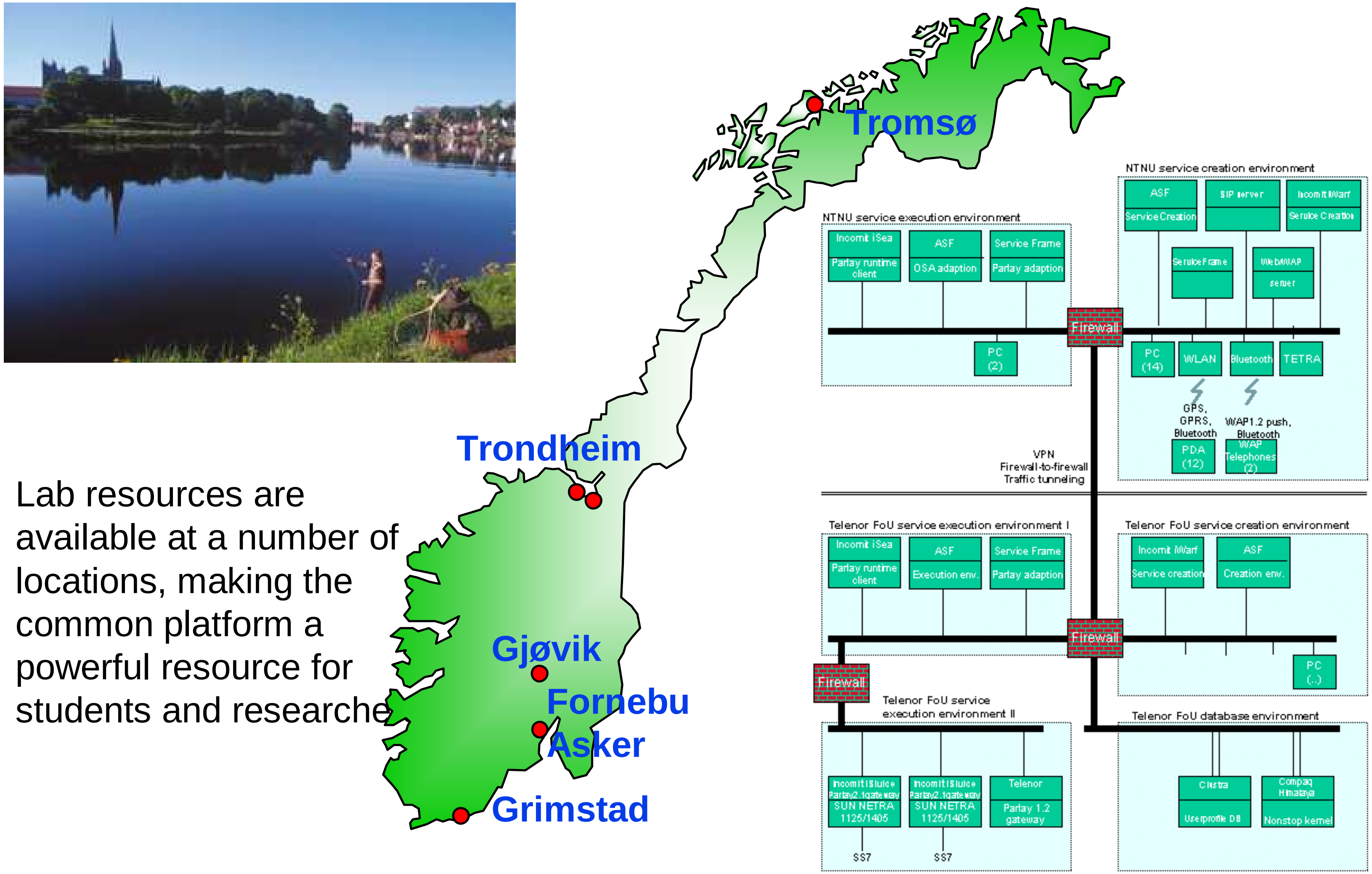
Provides UML2.0 concepts

Java VM

The emerging service context

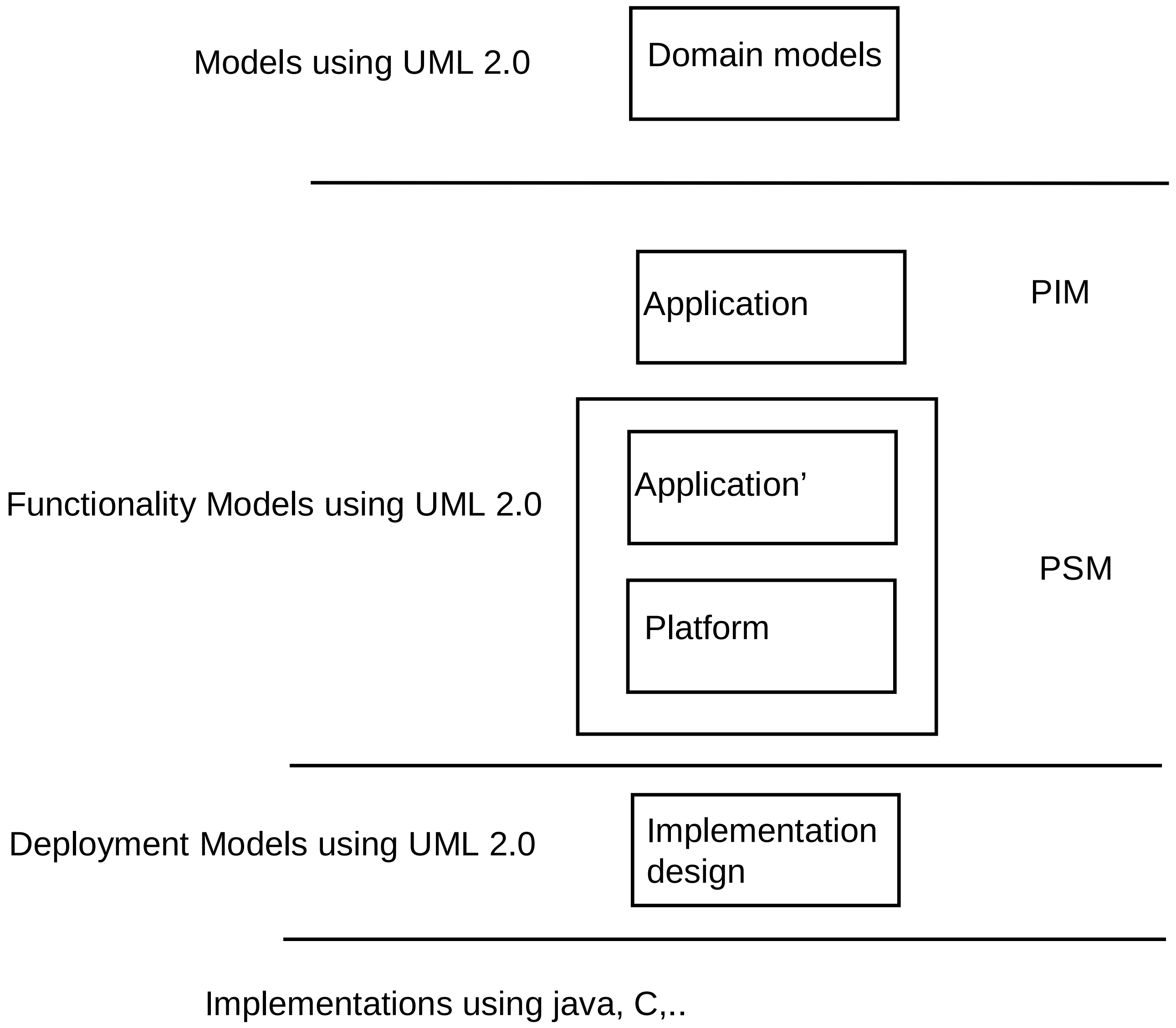


@work in a distributed environment



Lab resources are available at a number of locations, making the common platform a powerful resource for students and researche

Using the Model Driven Architecture



Models using UML 2.0

Domain models

Application

PIM

Functionality Models using UML 2.0

Application’

Platform

PSM

Deployment Models using UML 2.0

Implementation design

Implementations using java, C,..