THALES

www.thalesgroup.com


# Evaluating Enterprise Architectures through Executable Models

**15th ICCRTS**

*Evolution of C2: Where Have We Been? Where Are We Going?*

June 22-24 – Santa Monica, CA

N. Farcet & M. Ludwig

# Agenda

## Challenges & stakes

- Context
- Stakes
- Existing tools & methods
- Goals

## Our approach

- Architecture description language
- Qualitative architecture variant evaluation
- Quantitative architecture variant evaluation
- Tools

# CHALLENGES & STAKES

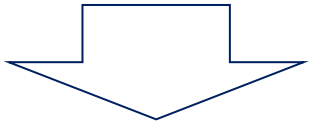
# Scope of the presentation

## Systems of Systems

- …*"a set of arrangement of systems that results when independent and useful systems are integrated into a larger system that delivers unique capabilities"*
  *[Defense Acquisition Guide Book]*

## Service-Oriented organizations

- *"Service Oriented Architecture (SOA) is a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains."*
  *[OASIS SOA Reference Model]*



*Enterprises* : networks of cooperating entities

# Challenges

## Understanding the key aspects of the enterprise architecture

- Despite an increasing complexity in organizations and service chains

## Being confident in its ability to fulfill its objectives…

- Identifying the key operational capabilities and mastering their availability
- Identifying and mastering the critical service chains
- Identifying and mastering the key resource flows

## ….in an unpredictable operational environment

- Handling unforeseen operational events and mission reorientations
- Enabling dynamic collaborations

# Existing tools to support architecting

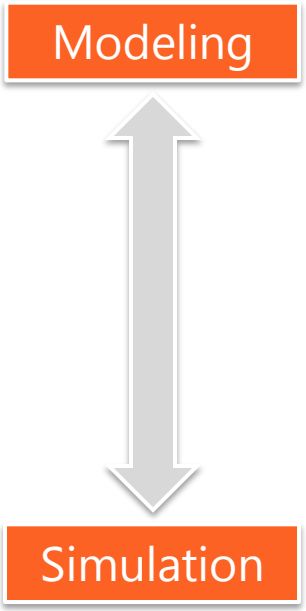
## Architecture Frameworks & associated tools

### Benefits

- Procurement-oriented
- Multiple viewpoints
- Standard-based
- Shared model-based reference

### Limitations

- Proprietary standard implementations
- Poor consistency check between views
- Limited or no evaluation means




## Simulation

- Tools: Proprietary technico-operational simulators, SIMUL8, ANYLOGIC, DGA DirectSim…

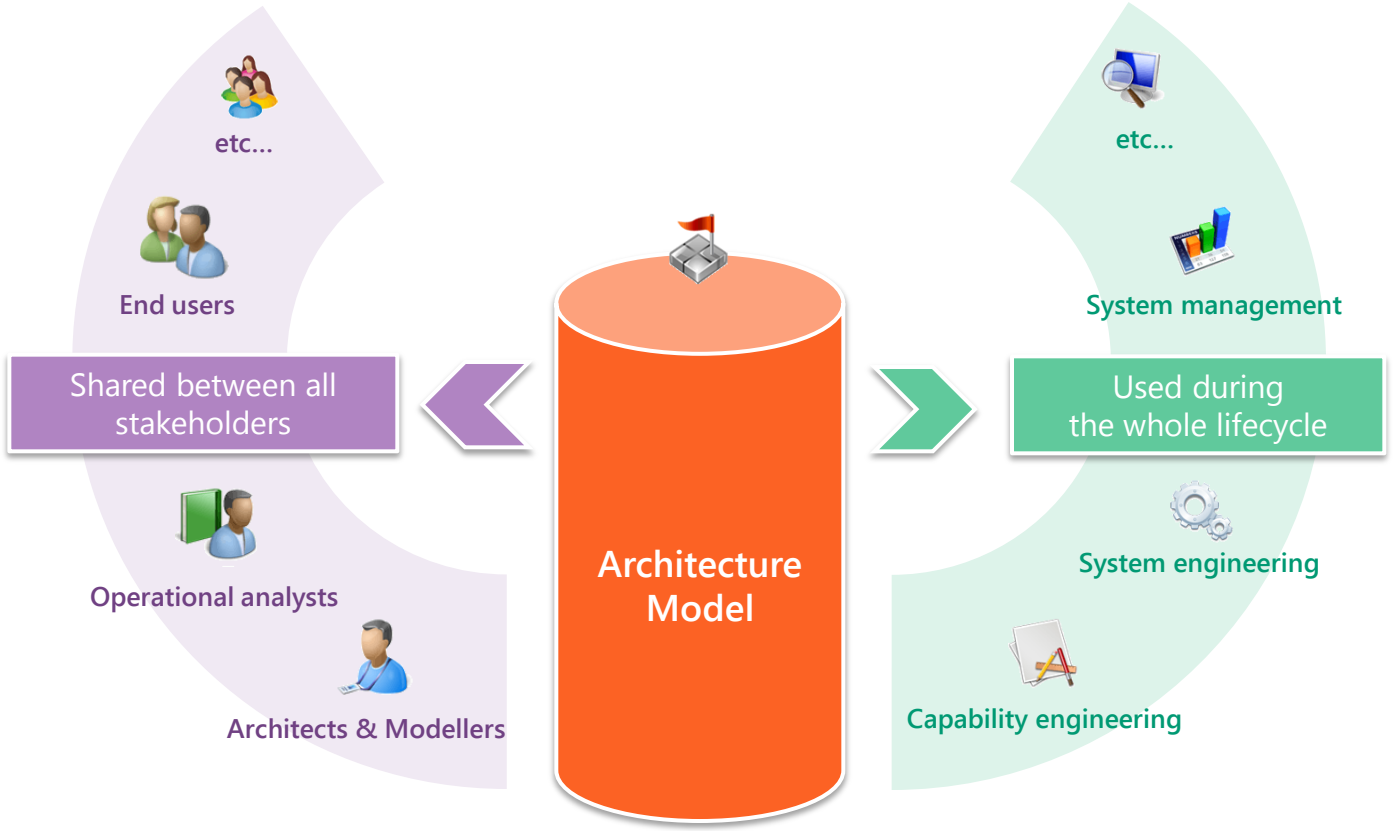
### Benefits

- Focused evaluation according to target SLA
- E.g. effectiveness, efficiency, robustness, sizing, deployment…

### Limitations

- Discontinuity with the modeling phase
- Can request a certain effort
- Not always architecture-centric (focused simulations)

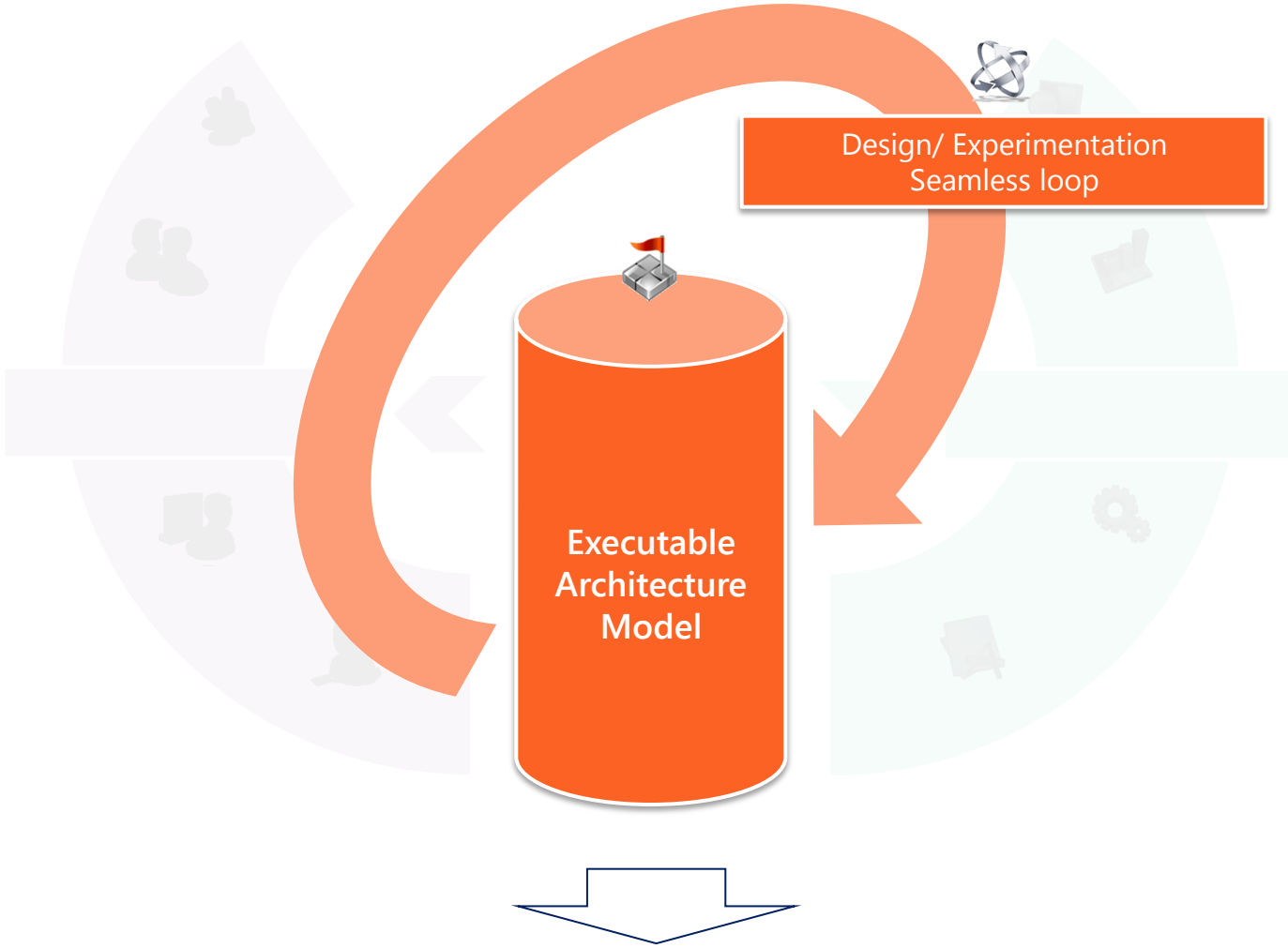
# Architecting based on a shared model




Building a **common vision** from **different points of view**

Sharing a **common reference** to analyze **different concerns**

# Architecture evaluation and incremental development through model execution



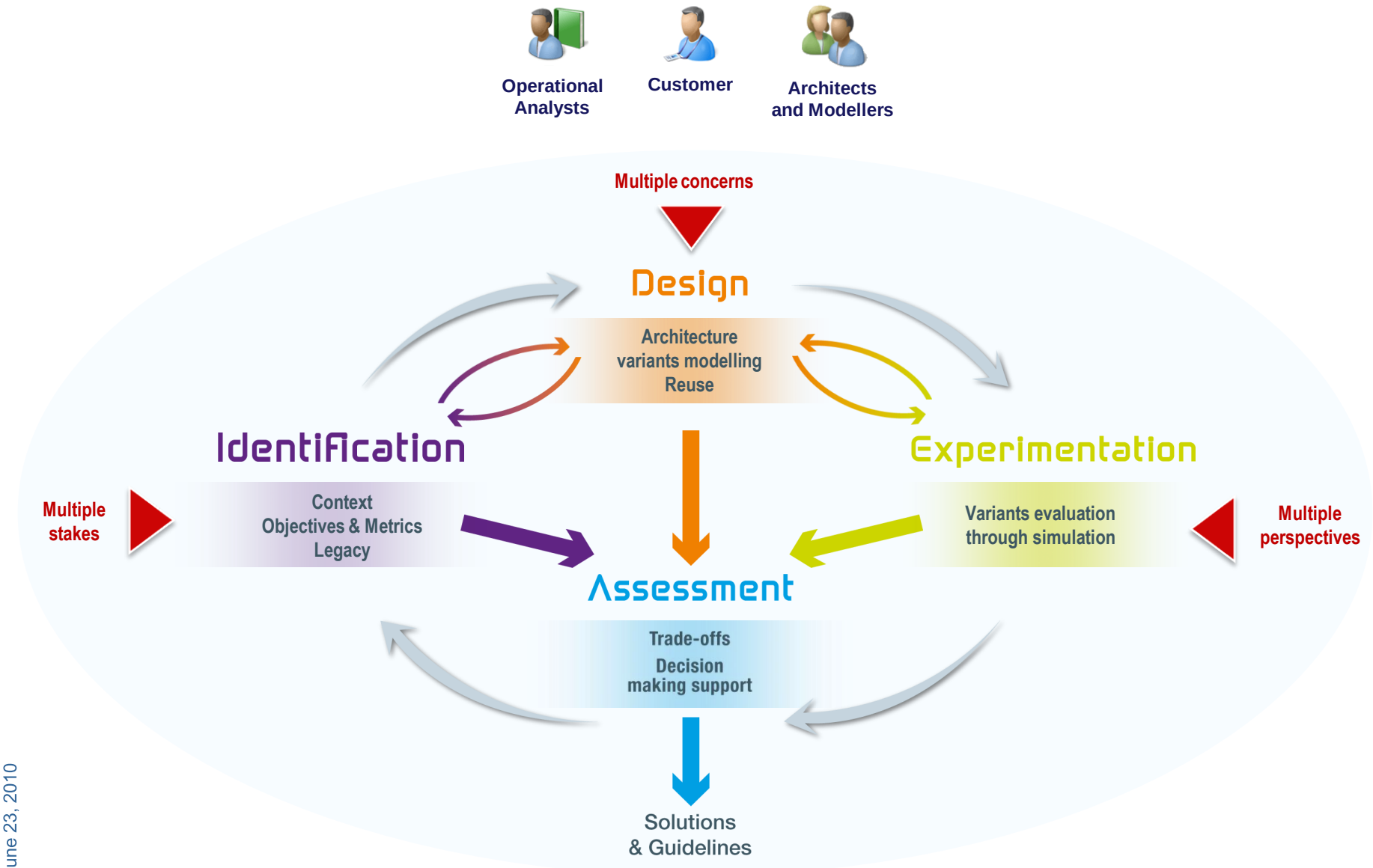

Enabling a rapid prototyping approach

# Typical SoS architecting issues

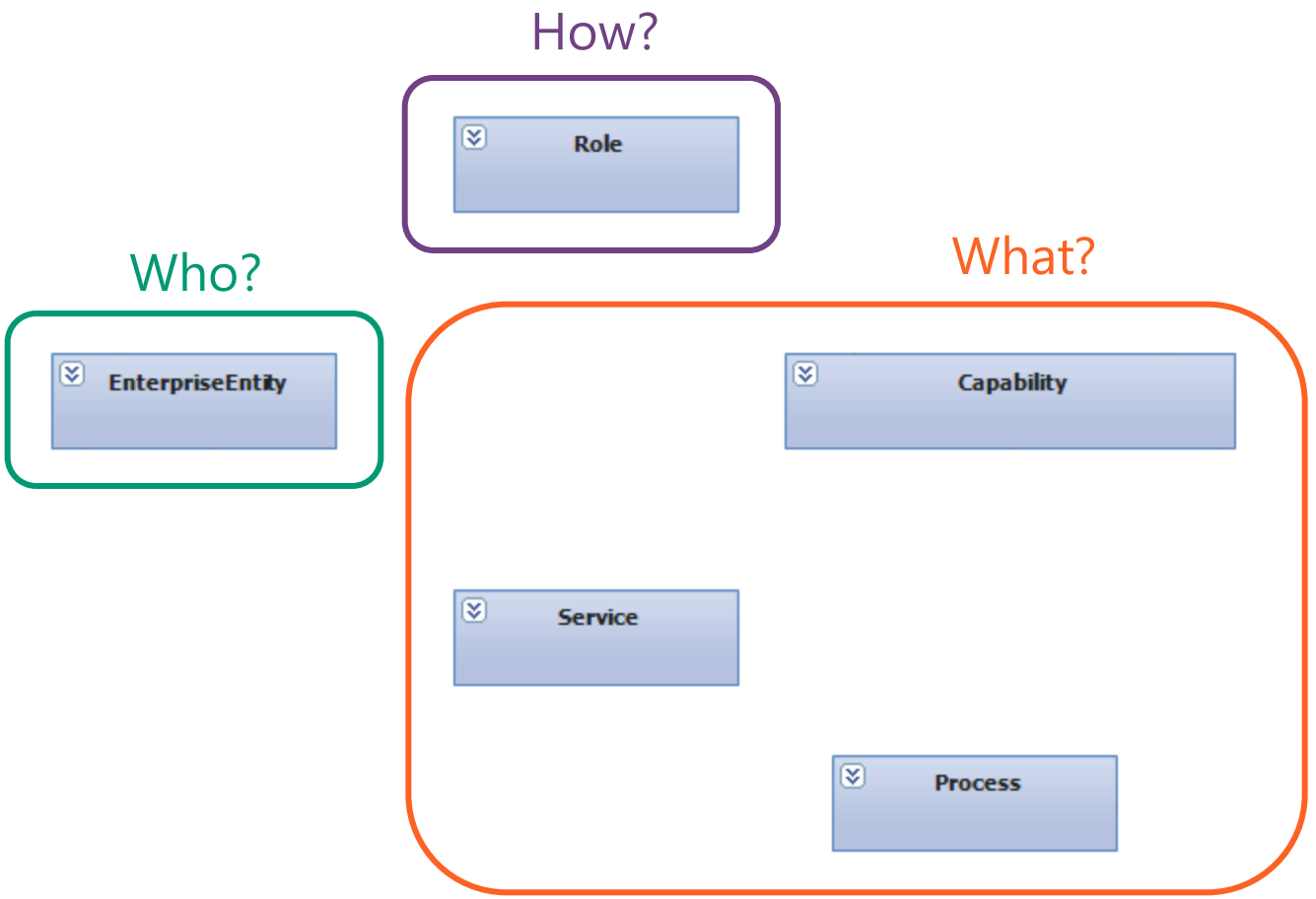
- Service chains and critical capabilities analysis
  - Identifying the critical service chains and the capabilities they rely on
  - Master durations and synchronizations
- Decision delegation and impact on possible removal of hierarchical levels
  - Collaborations vs. hierarchical command chains
  - Latency vs robustness of operation
- Information distribution and flows organization
  - Distributed vs. centralized architecture
    - Publication / subscription according to operational needs & communication constraints
  - Information availability at the edges
    - Fusion, filtering, routing, and caching algorithms
- Supervision, reconfiguration, and degraded modes management
  - Proper supervision information to the right actor
  - Autonomy areas *vs* hierarchical chains of command compromises
  - Radio silence and degraded modes management
- Logistics flows organization
  - Push *vs* on-demand logic
  - Sizing

# OUR APPROACH

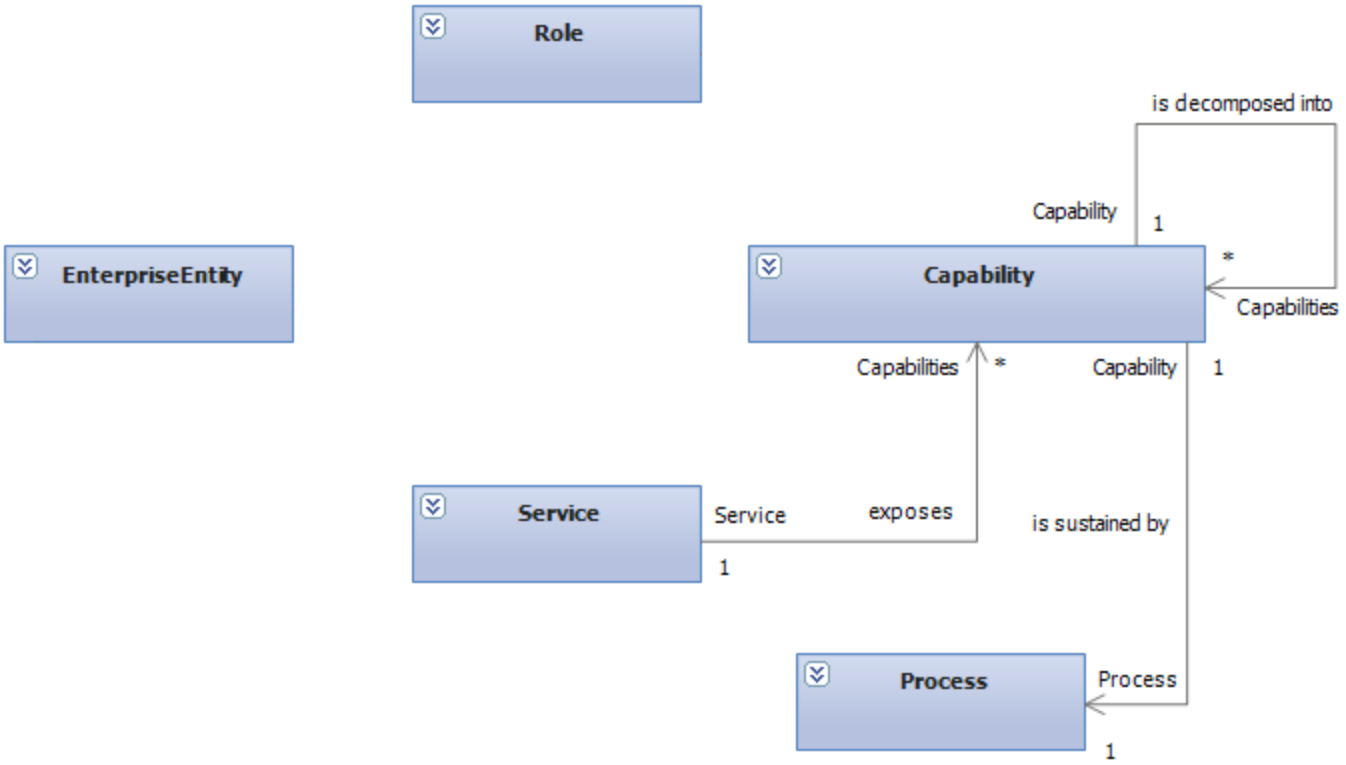
# The IDEA iteration



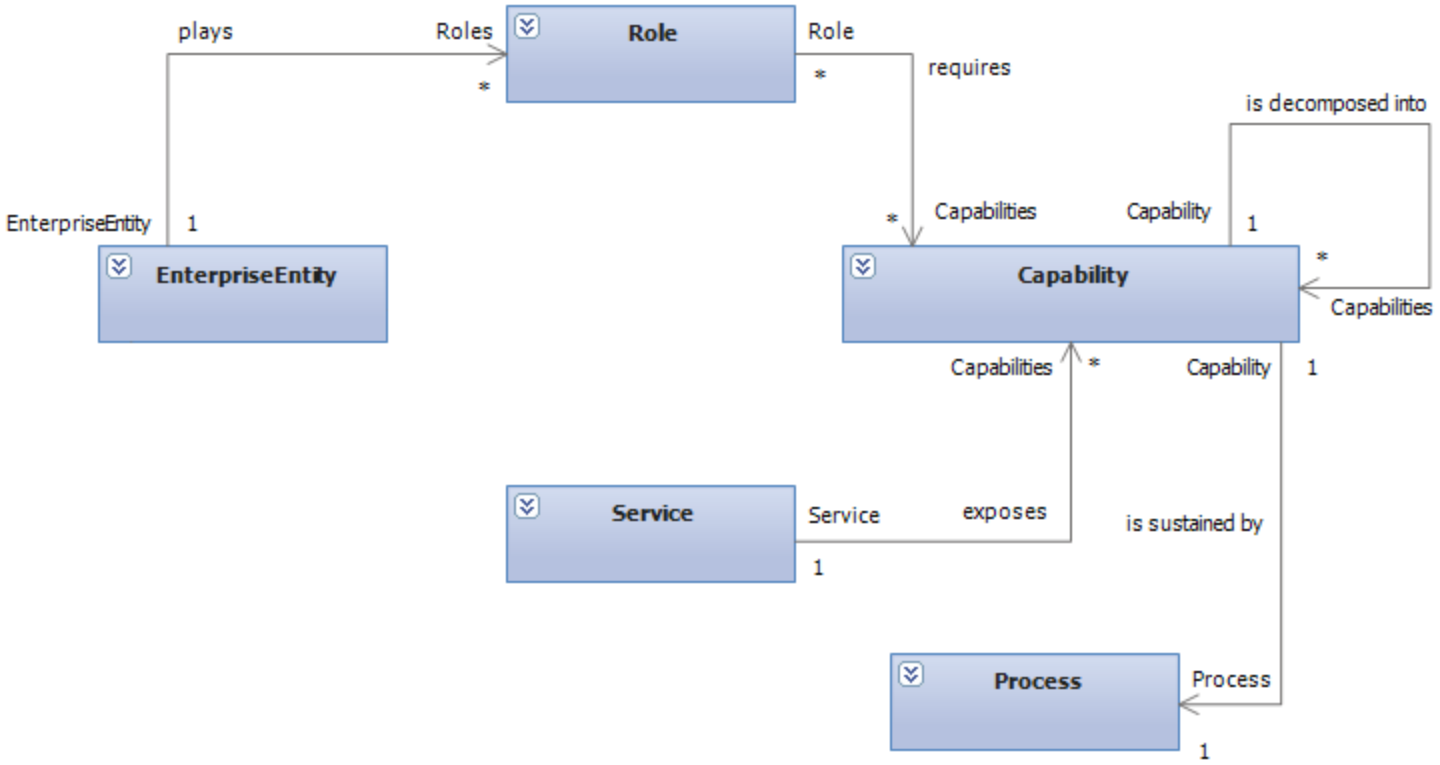
# IDEA Metamodel : core concepts



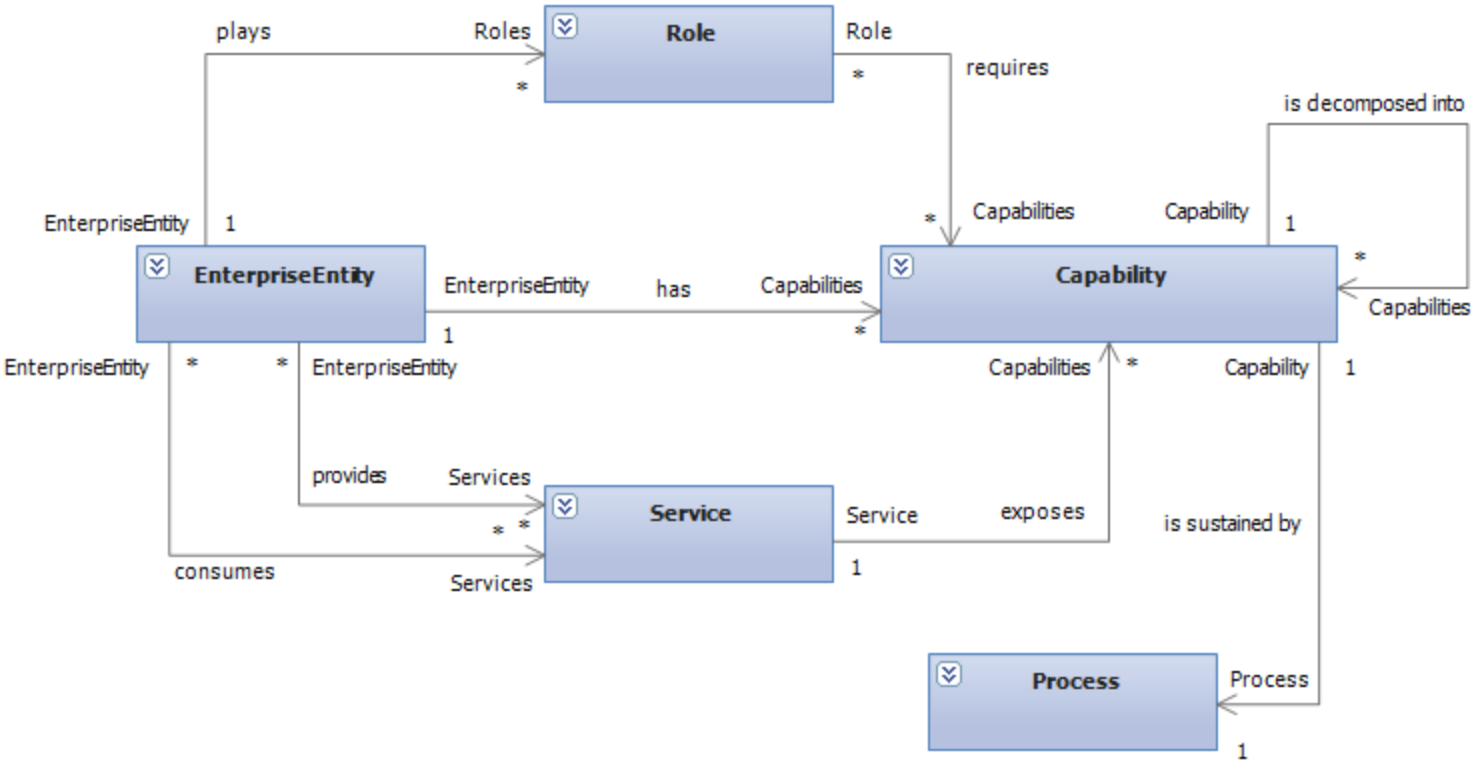
# IDEA Metamodel : core concepts



# IDEA Metamodel : core concepts



# IDEA Metamodel : core concepts



# Ensuring model executability

## Consistency rules (enforced)

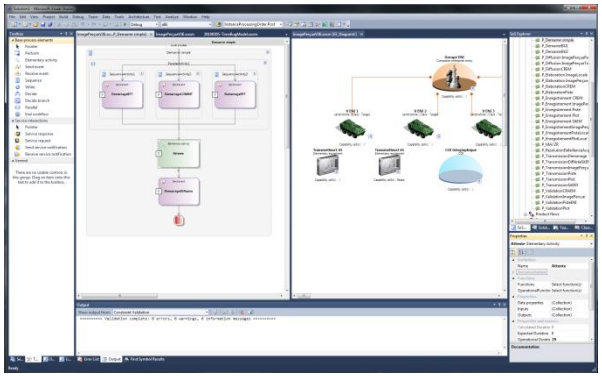
- Forbid the user to create incoherent (non executable) architecture patterns
- Propagates well-formness
  - E.g. Forbid two different Data elements to have the same name / Forbid to create a service interaction where the provider does not have the ability to provide the service

## Validation rules (on demand)

- Raise errors when the model is incomplete
- Do not prevent the model from being saved, but prevents it from being executed
  - E.g. Warn the user if an Entity requires a service but no one provides it

# IDEA Designer

- Multi-viewpoint approach to the creation of architecture models
- Static analysis (service chains robustness, end-to-end maximal duration...)
- Development environment for custom operational rules and measures of performance



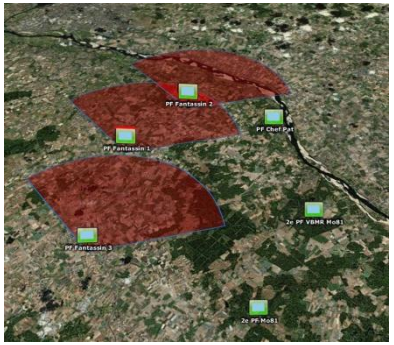
# IDEA Performer

- Deployed model execution
- Performance evaluation and logging
- (Evaluation through gaming)

# Architecture variant evaluation (1/2)

## Qualitative evaluation

- Instantiating entities and interactions in a sandbox or on a simulated theater of operations
- Running the processes in their operational context (current state of the entity, valued data sent by other processes...)



- Debug the model at all stages of its creation
  - Adopt an incremental creation of the model to help complex architecture understanding
- Conduct short and seamless execution / consolidation loops for domain relevance checking
  - Step by step execution of a process to evaluate the relevance of the way a process has been modeled,
  - Visualization of the interactions between deployed entities to evaluate the relevance of the way the information flows have been routed

# Architecture variant evaluation (2/2)

## Quantitative evaluation

- Logging performance during simulation, based on dynamic model elements properties
- Import into presentation and evaluation tool (Excel...)

- Identify the potential weak points of an architecture
  - E.g. identify roles that could lead to overloaded operators to redefine them or redistribute their activities,
  - E.g. identify probable bottlenecks in the processes or communication channels, ...
- Evaluate functional and non-functional metrics for variant comparison
  - E.g. compare the estimated traffic on various communication channels, ...

# Conclusions & perspectives

## Main point of the approach

- Using an executable enterprise architecture model to support rapid design-execution prototyping loops
  - To verify the conformity of the shared model with respect to all stakeholders' vision
  - To evaluate measures of performance that provide objective and comparable data for the evaluation of architecture variants

## Perspectives for our tool suite

- Improve the link with Architecture Frameworks
  - Current state : generation of a set of NAF views in Designer (beta)
  - Interoperability with NAF tools
- Improve the link with technico-operational simulation
  - To enable the planification of synchronized *rendez-vous* on the theatre
  - To improve the support of prediction of the impact of the loss of a resource
  - To support C2 decision with ''logistics-aware'' system management

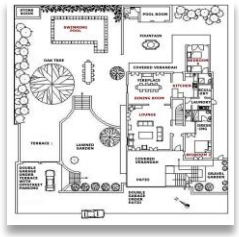
# QUESTIONS

# BACKUP

# Motivation for using Domain-Specific Languages

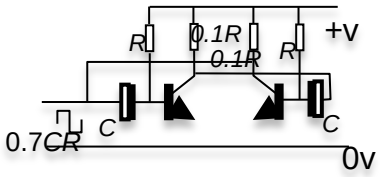
## Domain language engineering

- MDE benefits combined with a ''domain centric'' approach



## Key point : domain knowledge capture…

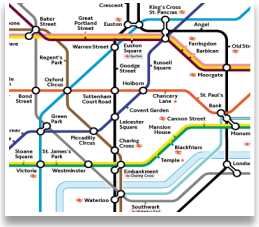
- Meta-models (abstract syntax constraints, rules…)
- Semantic (ontology, free text)




## …through which artefacts can be produced

(automatically or not)

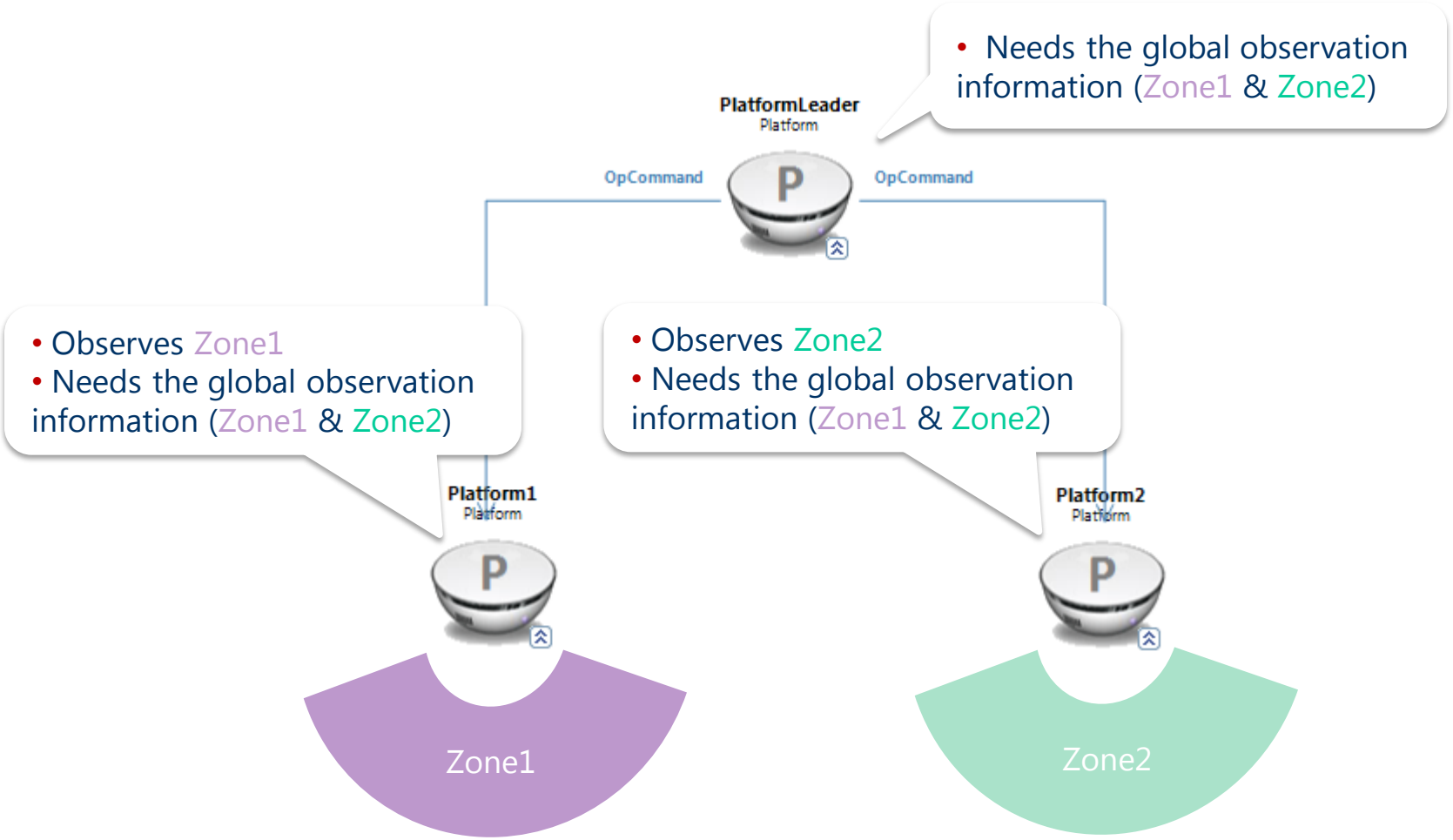
- Dedicated modeling notation & modeling tool
- Dedicated repository artefacts (navigation, checks…)
- Domain rules & constraints checks (at design- and runtime)



Adapted from J.M. Prieur

# An example : Information Fusion (1/4)

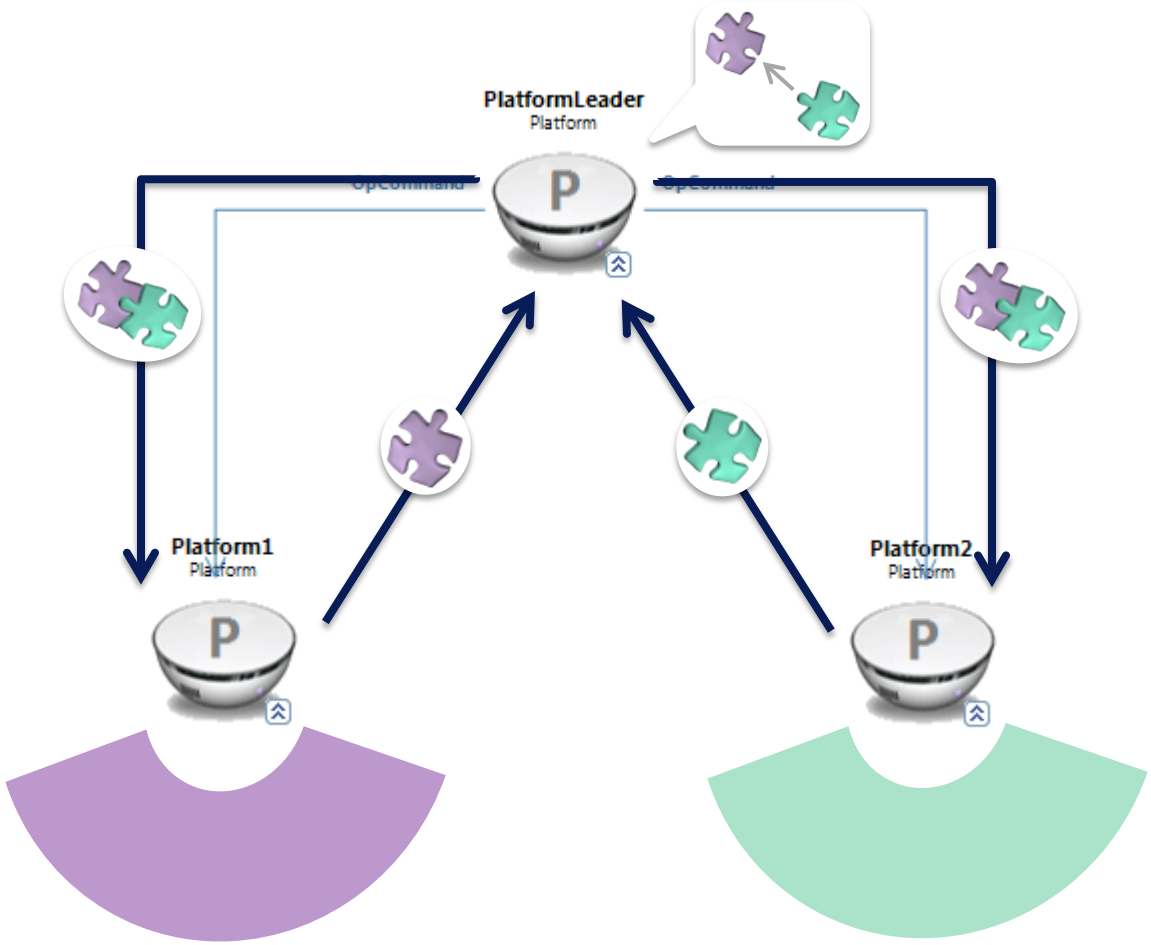
Let's consider an elementary Information Fusion group…



# An example : Information Fusion (2/4)

## Information flows variant 1 : along the Command chain

- *PlatformLeader performs the fusion and broadcasts the result*




Information flow

Observation information about Zone1

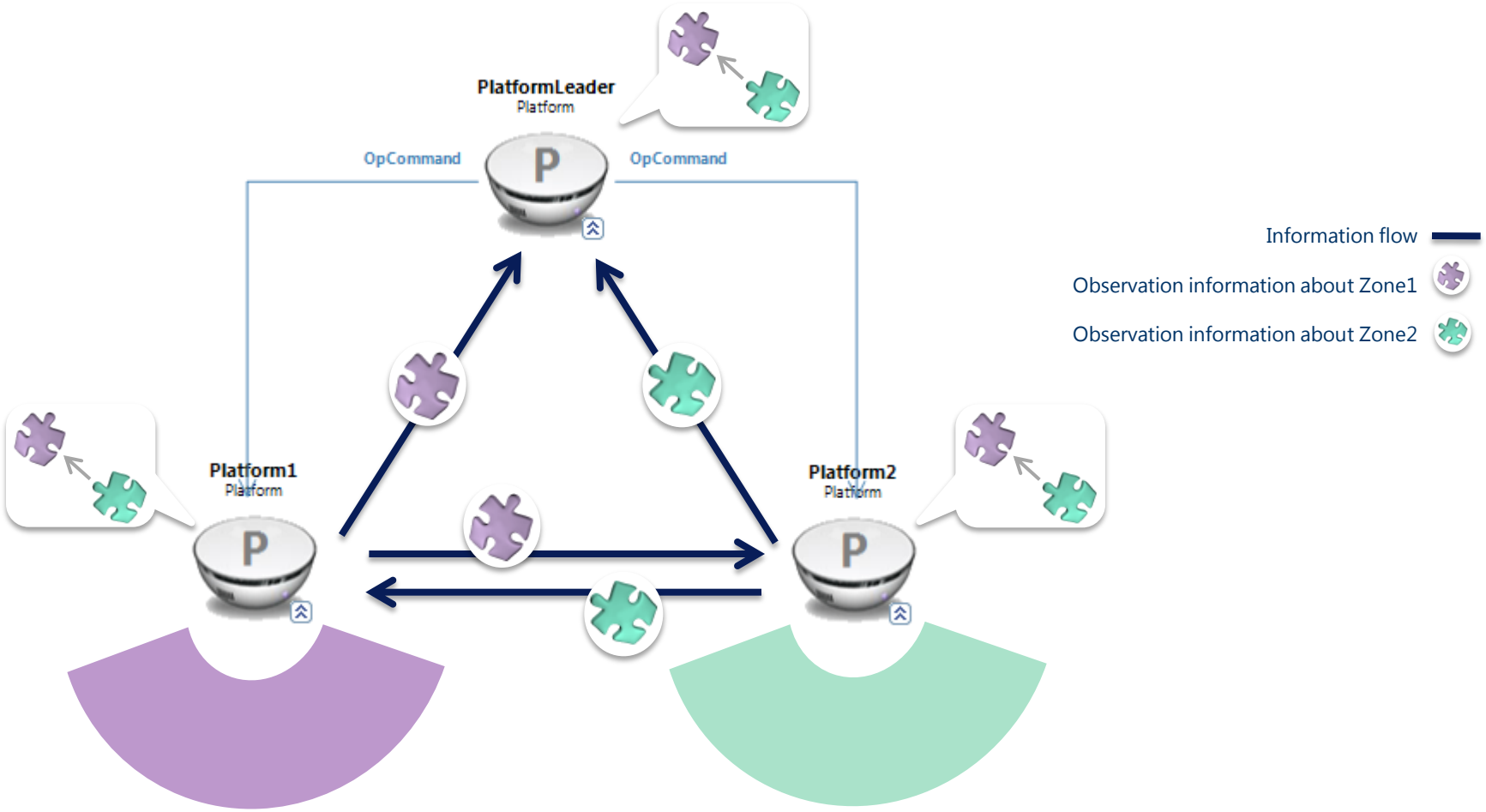
Observation information about Zone2

Global (fused) observation information

# An example : Information Fusion (3/4)

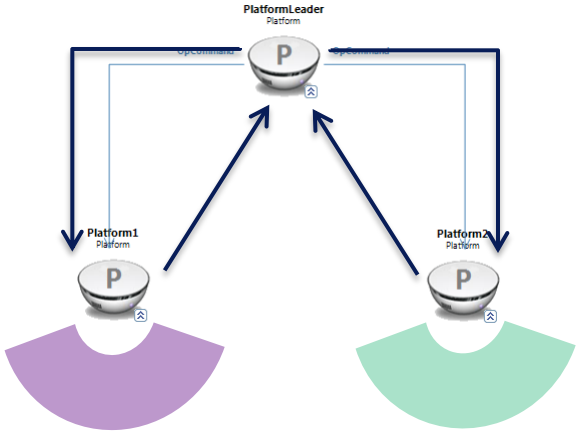
## Information flows variant 2 : along and across the Command chain

- *Each platform performs a local fusion of observation information*



# An example : Information Fusion (4/4)

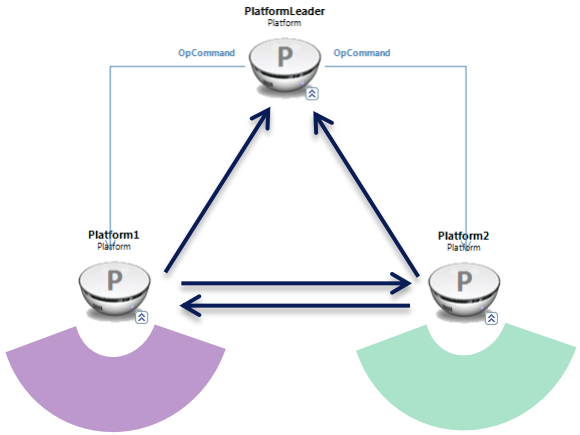
## Variant 1




- \+ All platforms have the same global information
- \+ Less load for Platform1 and Platform2
- \- Latency
- \- Single point of failure (PlatformLeader)

How much?

## Variant 2




In which case(s)?

- \- Possible coherence problems between all global informations
- \- More CPU load for Platform1 and Platform2

How much?

- \+ Latency
- \+ Redundancy (robustness)

An executable architecture model allows conducting the quantitative analysis necessary for an objective evaluation and comparison of these two variants.