Model Driven Engineering

joel.champeau@ensta-bretagne.fr

Abstraction and modelling

Abstraction definition [Delaunay-EncUniv] :

- Simplification: simplifying the reality considered too complex (apprehending in a unitary way and leading to a law)
- Generalization: generalize to conceptualize
- By selection: operation preparatory to classification
- Schematics: for modelling and its formalization for analysis

System and Software Modelling

- Our Models are really a system abstraction generally based on Generalization, Selection and Schematics
- Many definitions of Model in the literature MDE […….]

A Model: Set of abstractions semantically defined and relative to a selected intention

Abstraction and modelling

Used of Models

- Many needs to satisfy:
  - Static and dynamic description
  - Present and understand
  - Specification, Design…,
  - AnalysiS
- Many Intentions associated to the models
- Many concerns to take into account for each intention

Generalisation, a Model for:

- Descriptive et Prescriptive [Sei03]
- Explanatory & Constructive [At11,Ku16]

A model for its ontological or prescriptive virtues

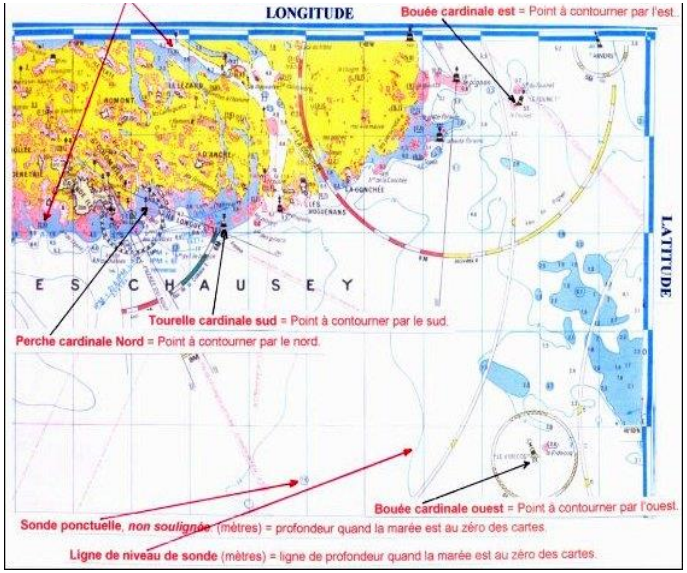
Model definition and metamodel

Providing an abstract definition of modeling languages

- Reflexive approach applying to the modeling language definition
- A model as a model definition = a Metamodel is a model of the model language features (concepts, concept properties, relationships)
  - Easy to understand
  - Tool implementation independent
- a Metamodel is a model of the model language features
  - Including concepts, concept properties, relationships and constraints

Metamodel definition

Model and meta-model : sea maps.



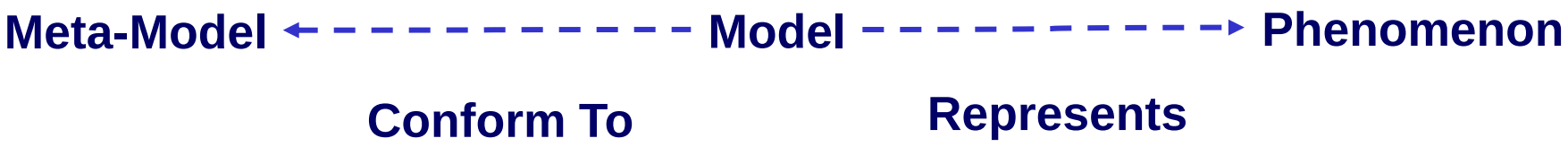
Meta-Model ← - - - - - - - - - - **Model** - - - - - - - - - - → **Phenomenon**

Conform to **Represents**

Metamodel definition

Model and meta-model :

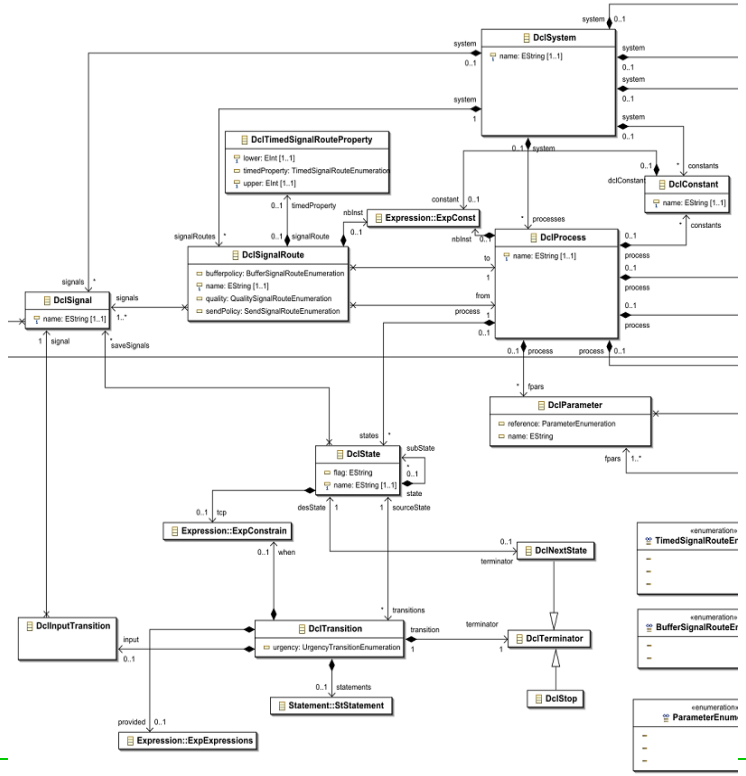
- Model represents the real world with a particular intention.
- Meta-model gives the definition of the modelling language. It's a model too.



Meta-Model :

a model described with a diagram of classes.

- Examples : definition of the modelling language IF for verification. (ENSTA Bretagne)



Conform to

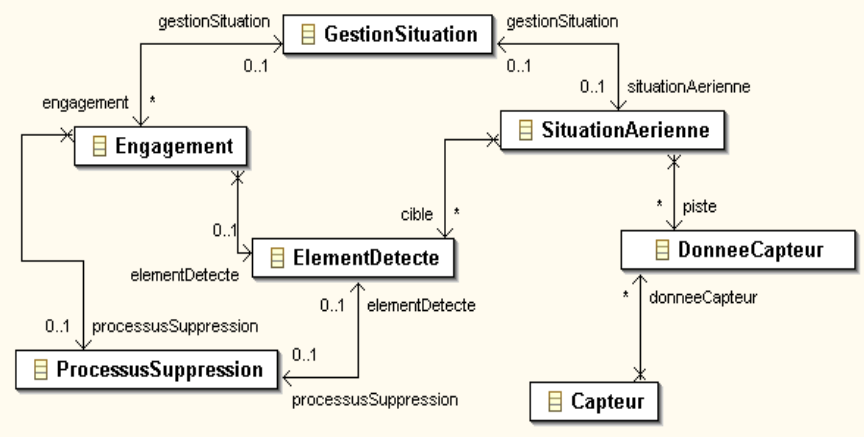
```
process transmitter(1);
var t clock;
var b boolean;
var c boolean;
var m data;

state start #start ;
    task b := false;
      nextstate idle;
endstate;
state idle;
  input put(m);
    output sdt(m, b) via {tr}0;
    set t := 0;
      nextstate busy;
endstate;
state busy;
  input ack(c);
```

Meta-Model :

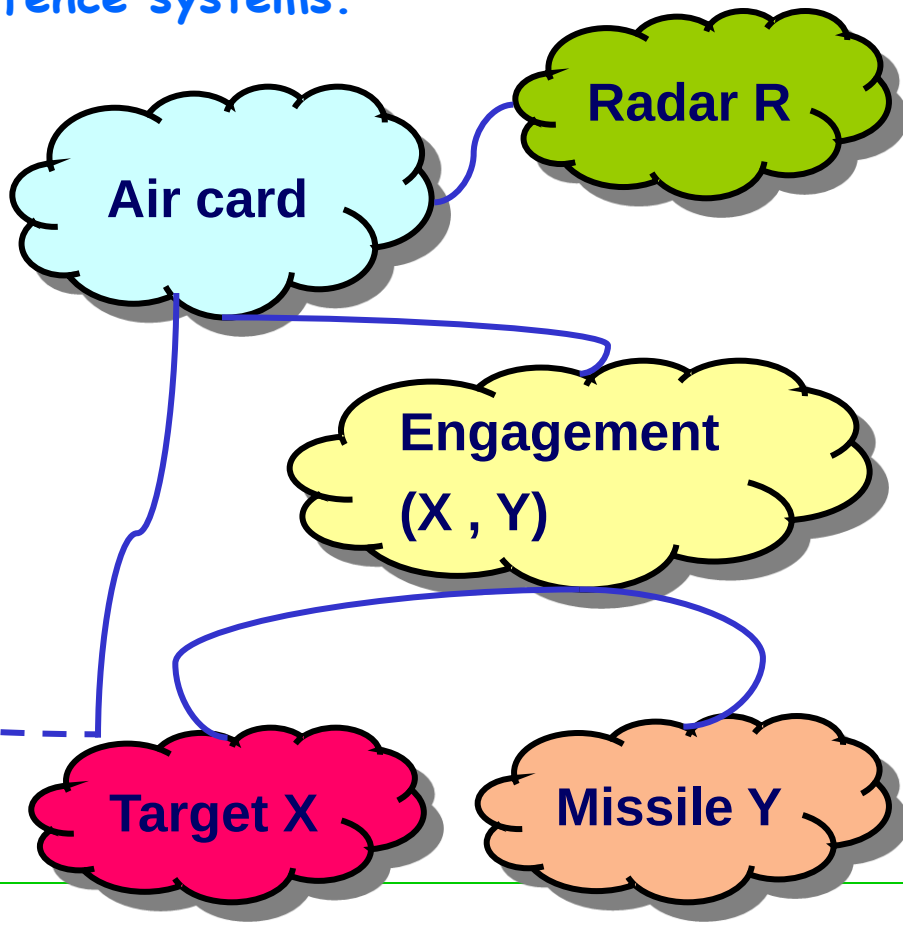
Meta-Model : Abstract model of a domain model

- Example : DSL for anti-air defence systems.



Meta-Model

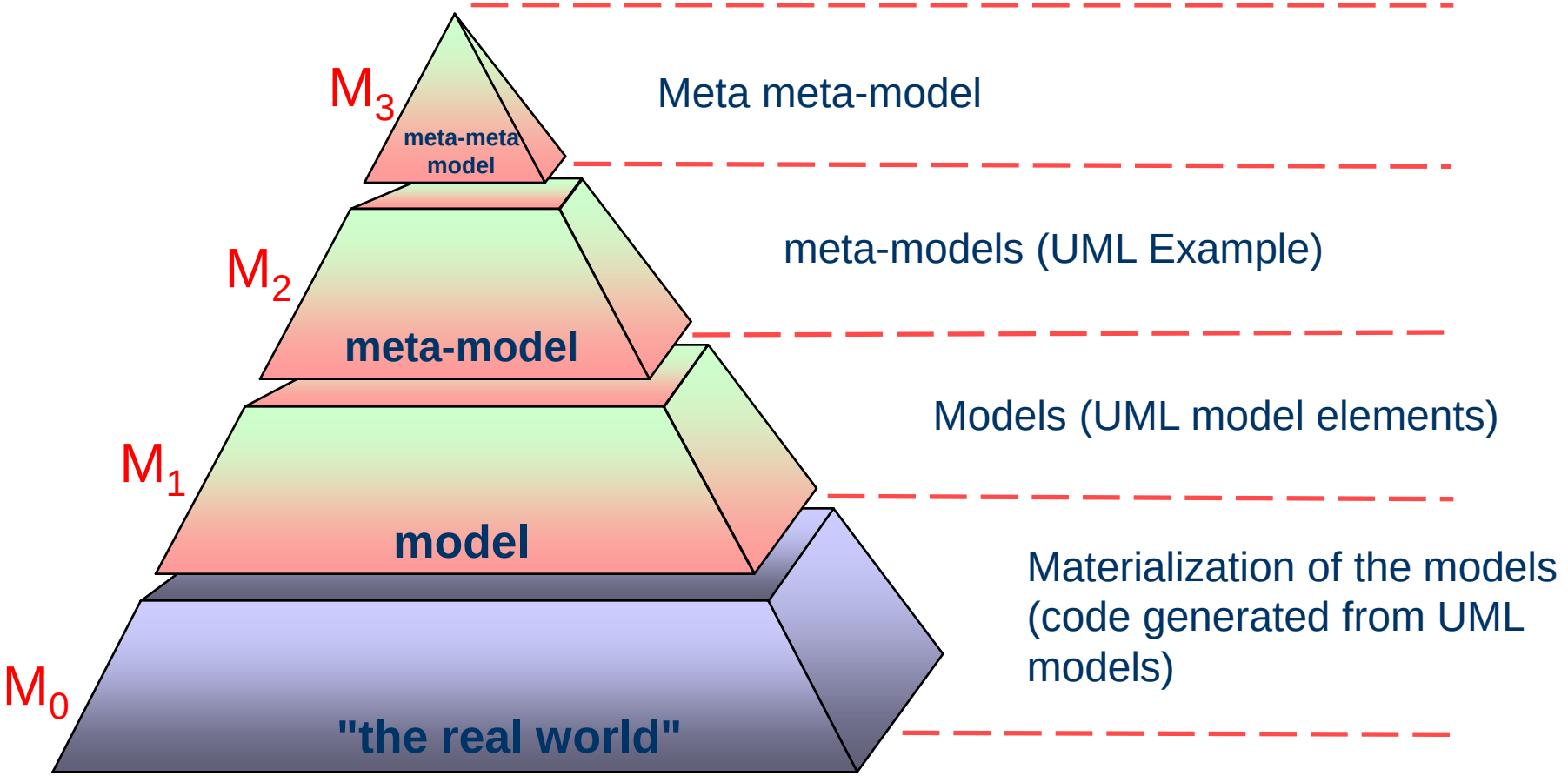
Conform to



Definition

- Metamodel = class model of concepts
  - Semantics of the language domain = Concept identification
  - Structural definition.
  - Concept properties:
    - Attributes to qualify the concepts
    - Operation to specifiy the concept behavior
  - Relationships between concepts
    - Role identification (role naming)
    - Composition or simple association
    - Cardinality
    - Generalization
  - Constraints on the previous structure description
    - Used of OCL, for example

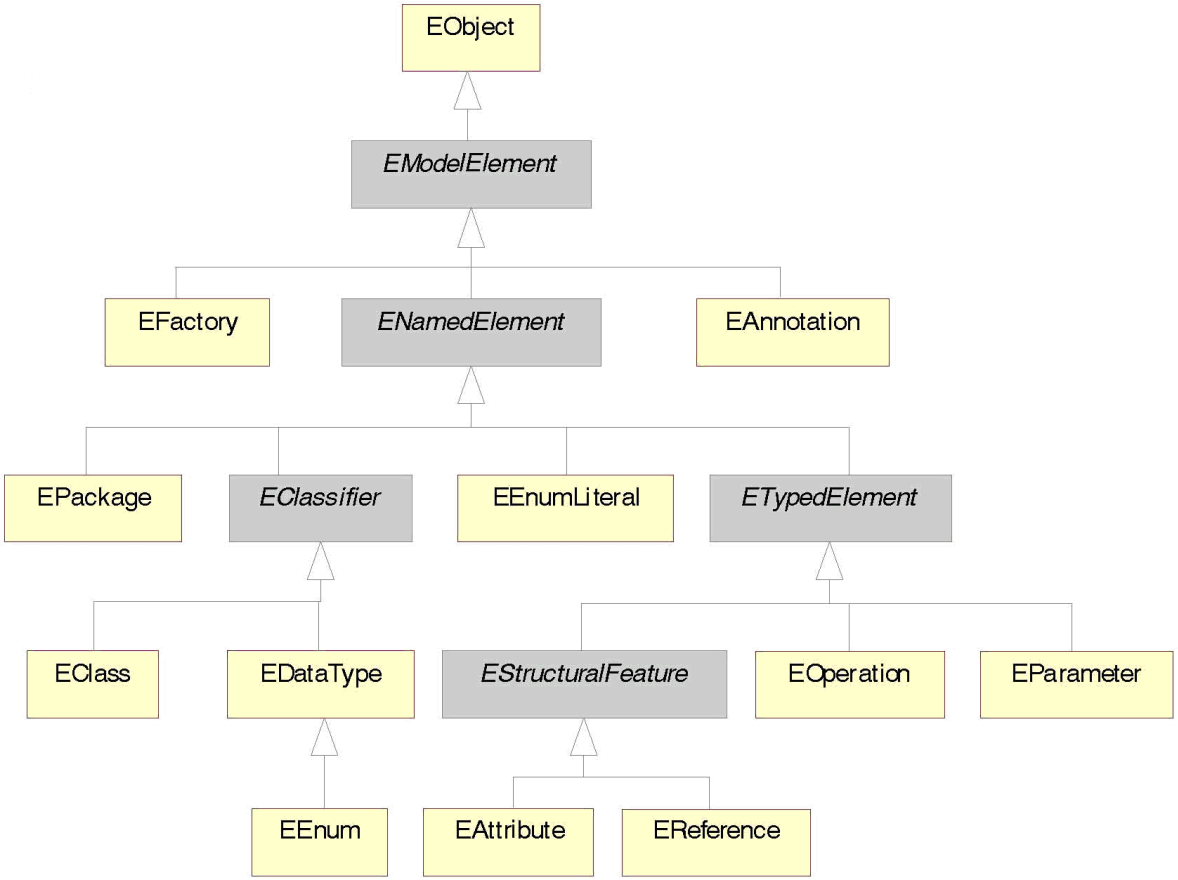
4 level paradigm



Meta-model description

MM description with a dedicated metamodel (meta-metamodel).

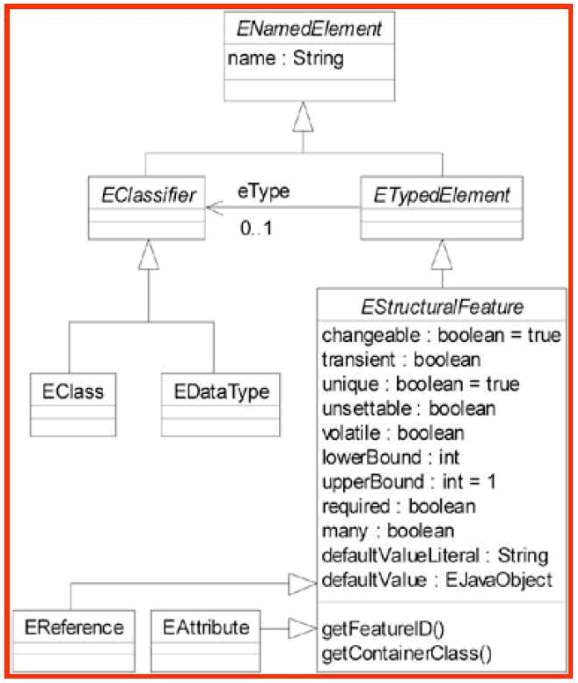
- Example of Ecore model.
  - Meta meta-model oin EClipse framework (EMF Eclipse Modeling Framework)
  - Ecore becomes a standard....

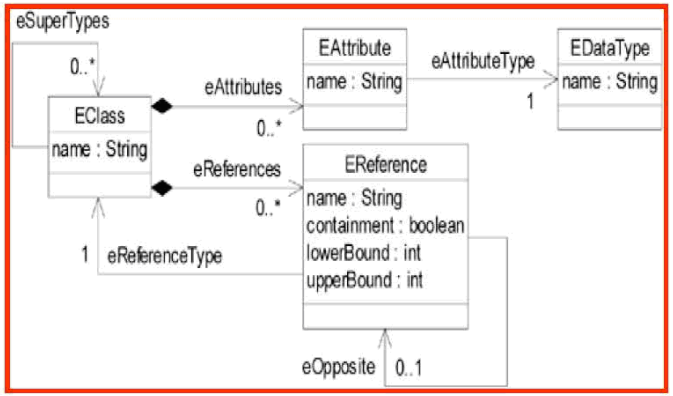


Meta-model description

MM description with a dedicated metamodel (meta-metamodel).

- Example of Ecore.

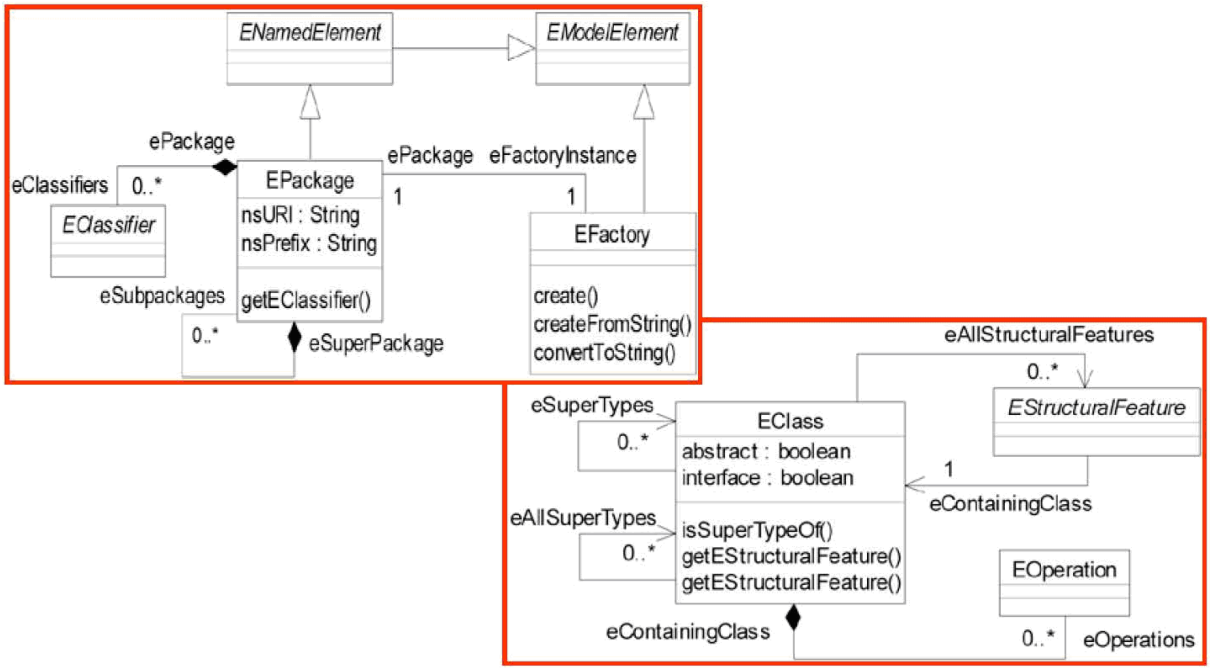




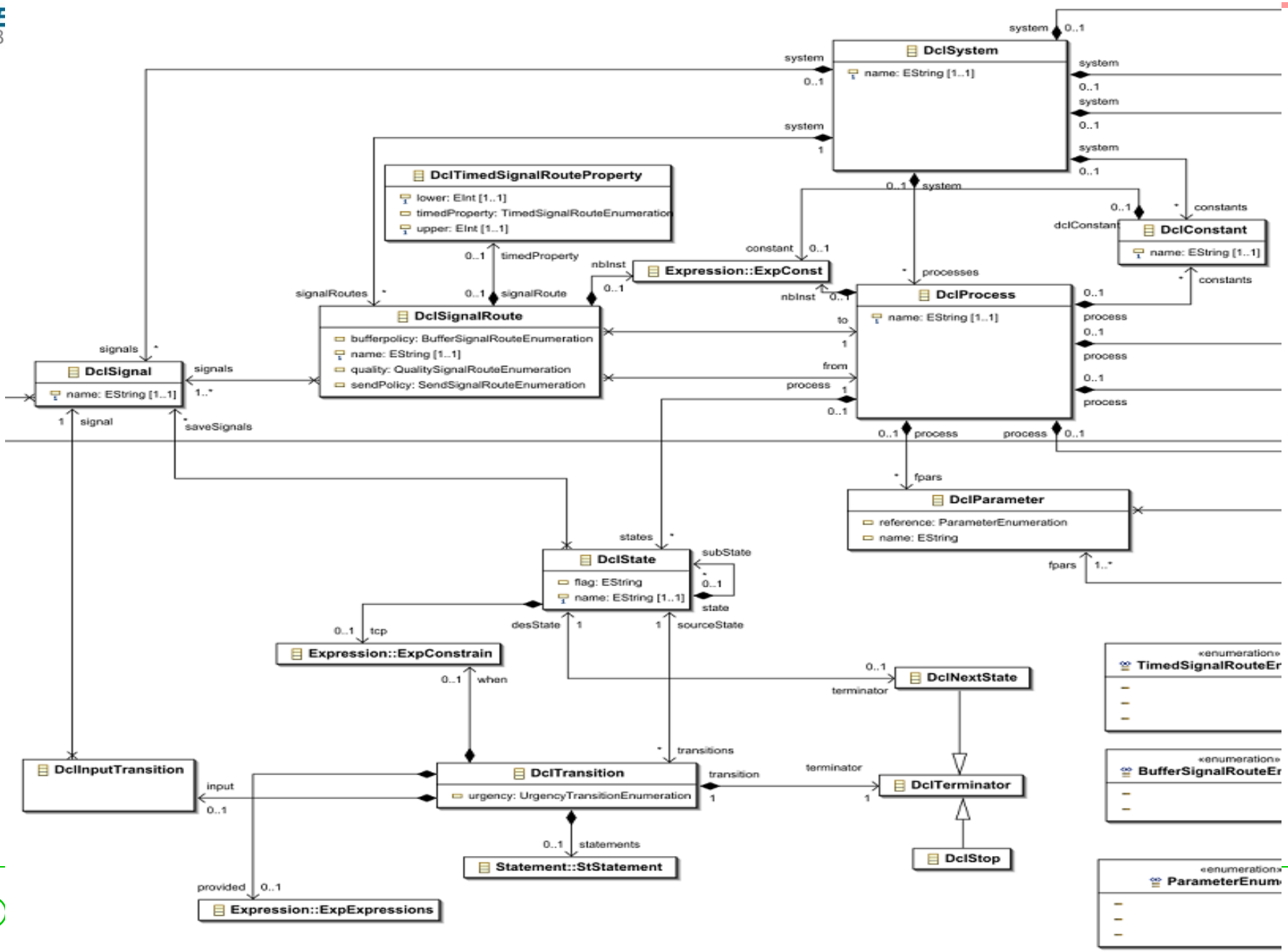
Meta-model description

MM description with a dedicated metamodel (meta-metamodel).

- Example of Ecore.



Metamodel definition

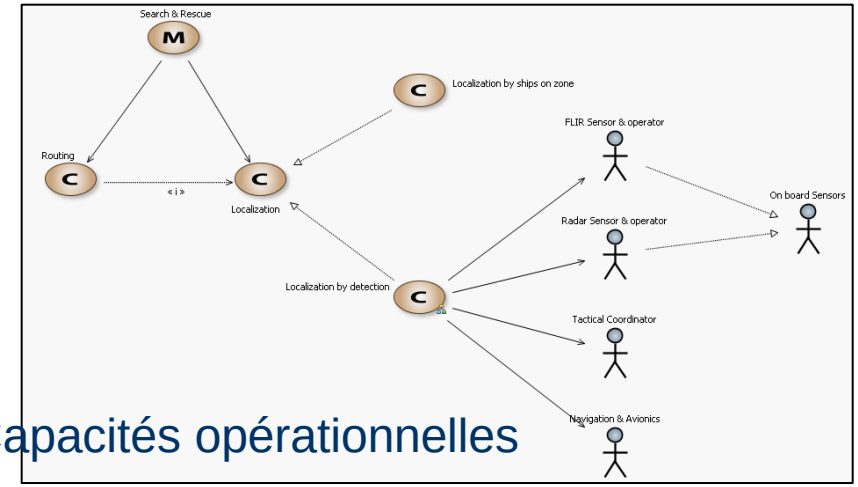


Software models and System models

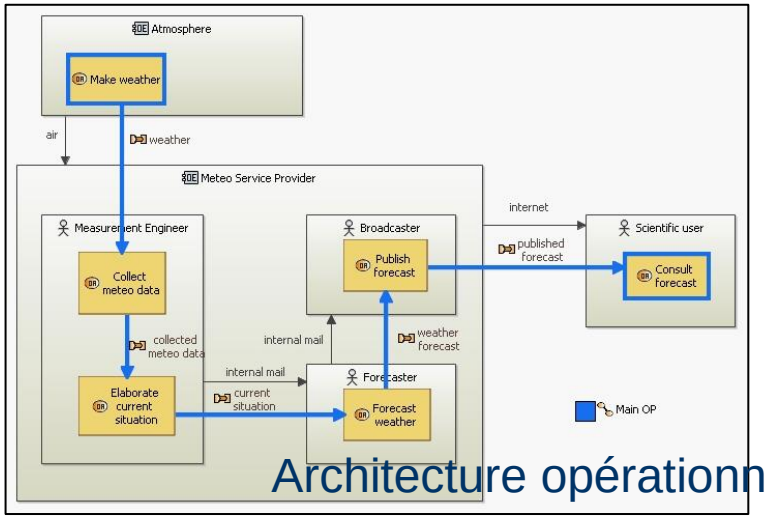
Some examples

- System level
  - General purpose : SysML (UV5.8)
  - Domain specific : Capella (Thales, UV5.8), NAF (Nato Architecture Framework, DGA)

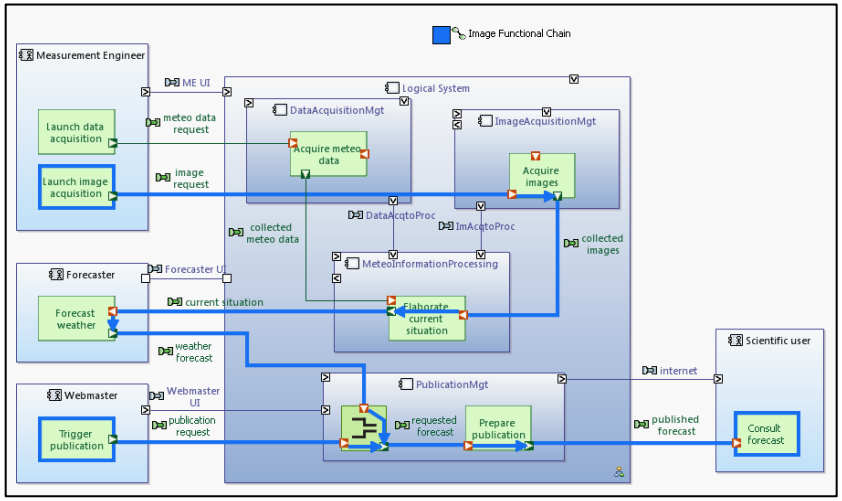
From Thales to Open source approach



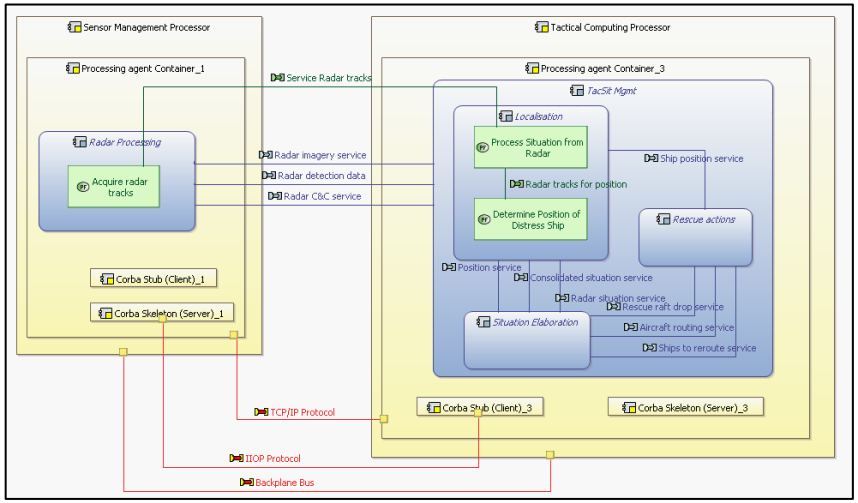
Capacités opérationnelles



Architecture opérationnelle

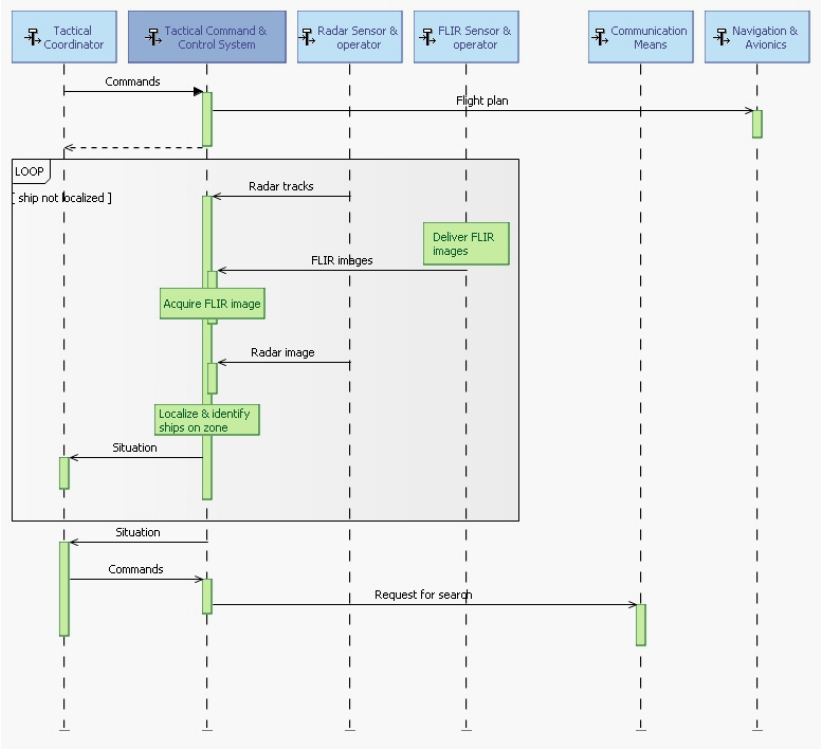


Architecture logique

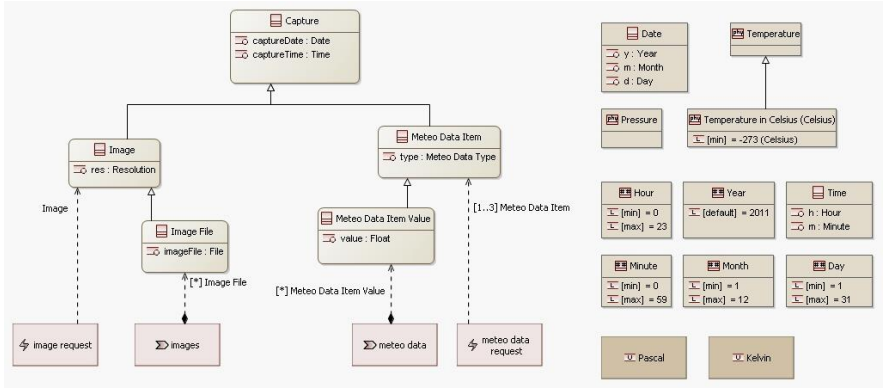


Architecture Physique

From Thales to Open source approach



Scénario fonctionnel



Modèle de données : Diag de classe

Software models and System models

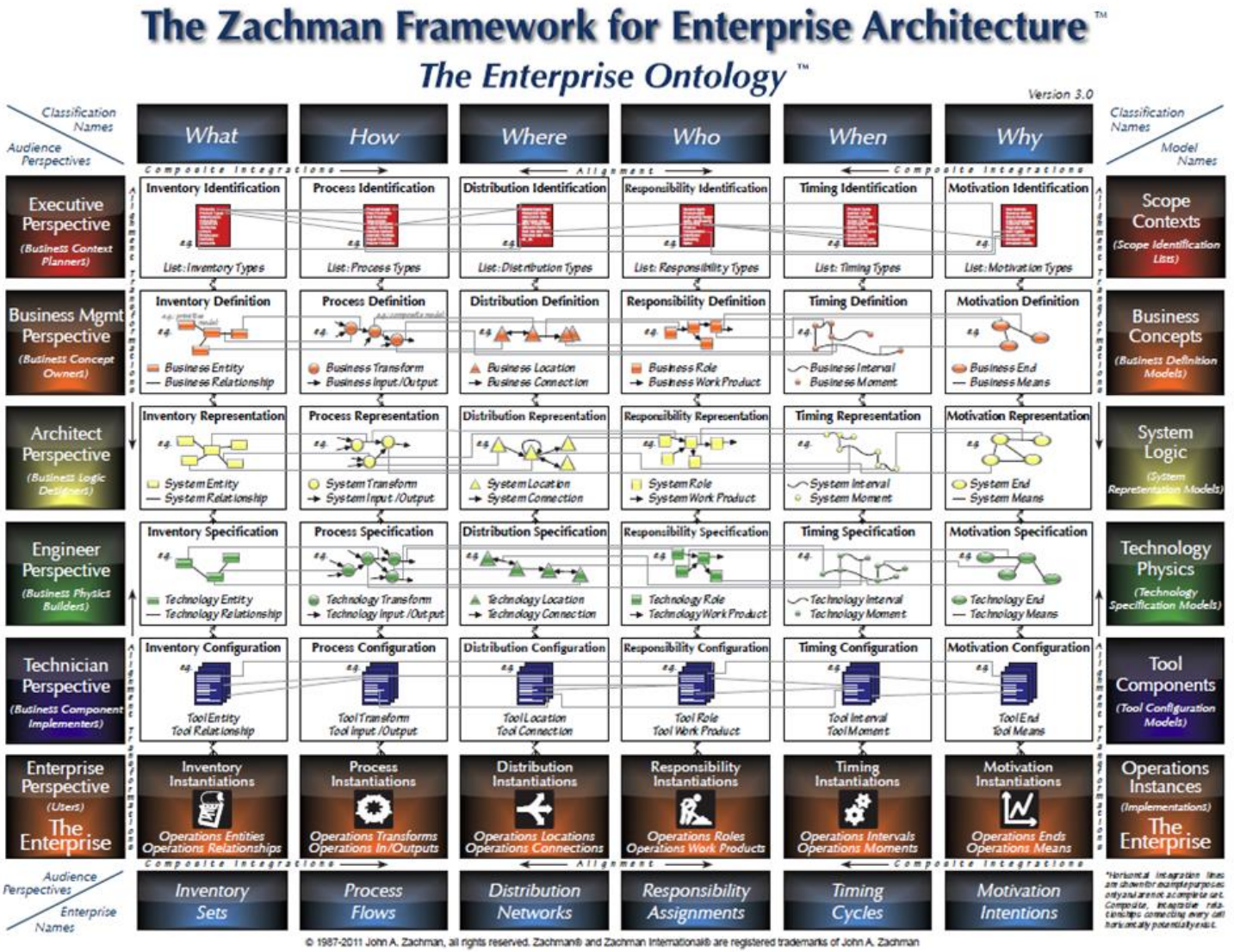
Some examples

- System level
  - General purpose : SysML (UV5.8)
  - Domain specific : Capella (Thales, UV5.8), NAF (Nato Architecture Framework, DGA)

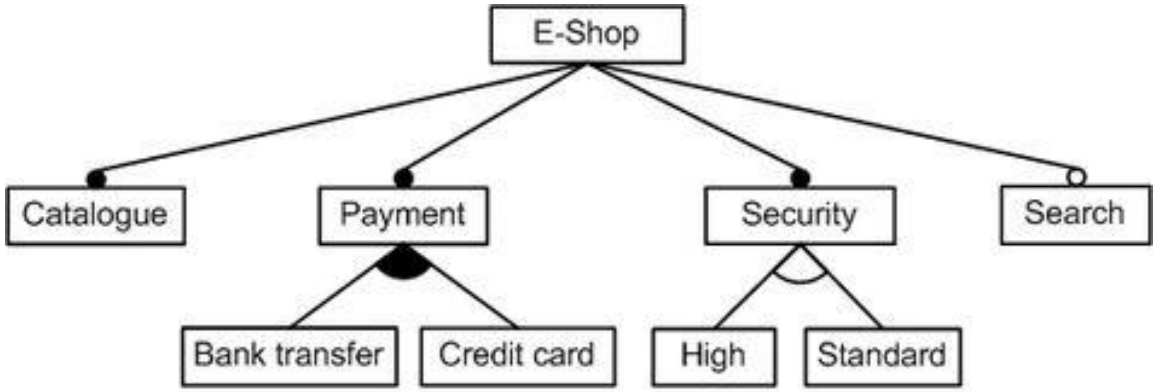
- Domain specific modeling
  - Entreprise Architecture : Zachman framework (IT), etc…
  - Design modeling : AADL (component modeling), UML, etc…
  - Software modeling : UML, Merise
  - (Software) Product Line : Feature modeling
  - DSL general Approach : Sirius (Eclipse foundation – Obeo)

Zachman framework

And TOGAF, etc…

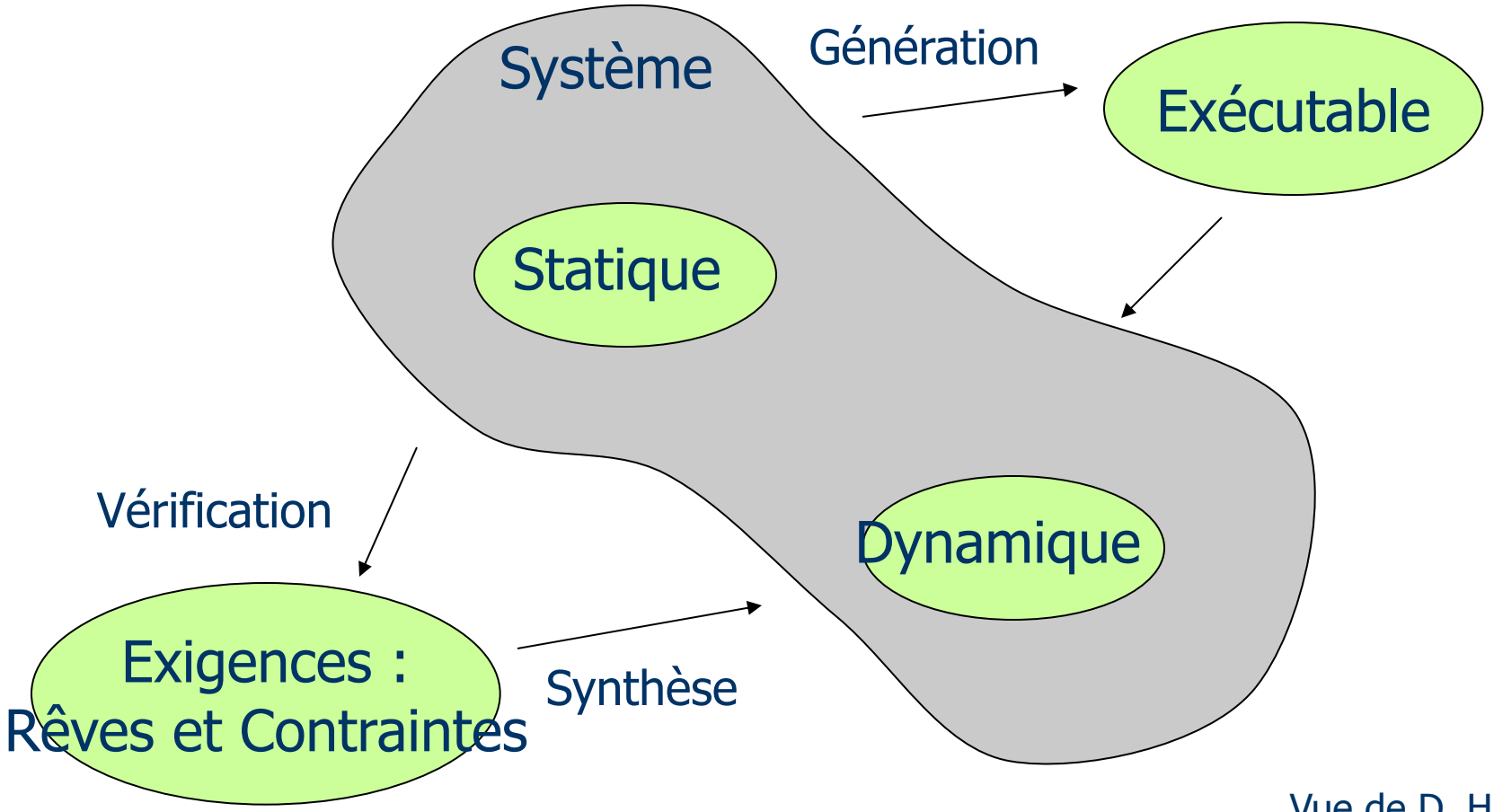


From system to software components



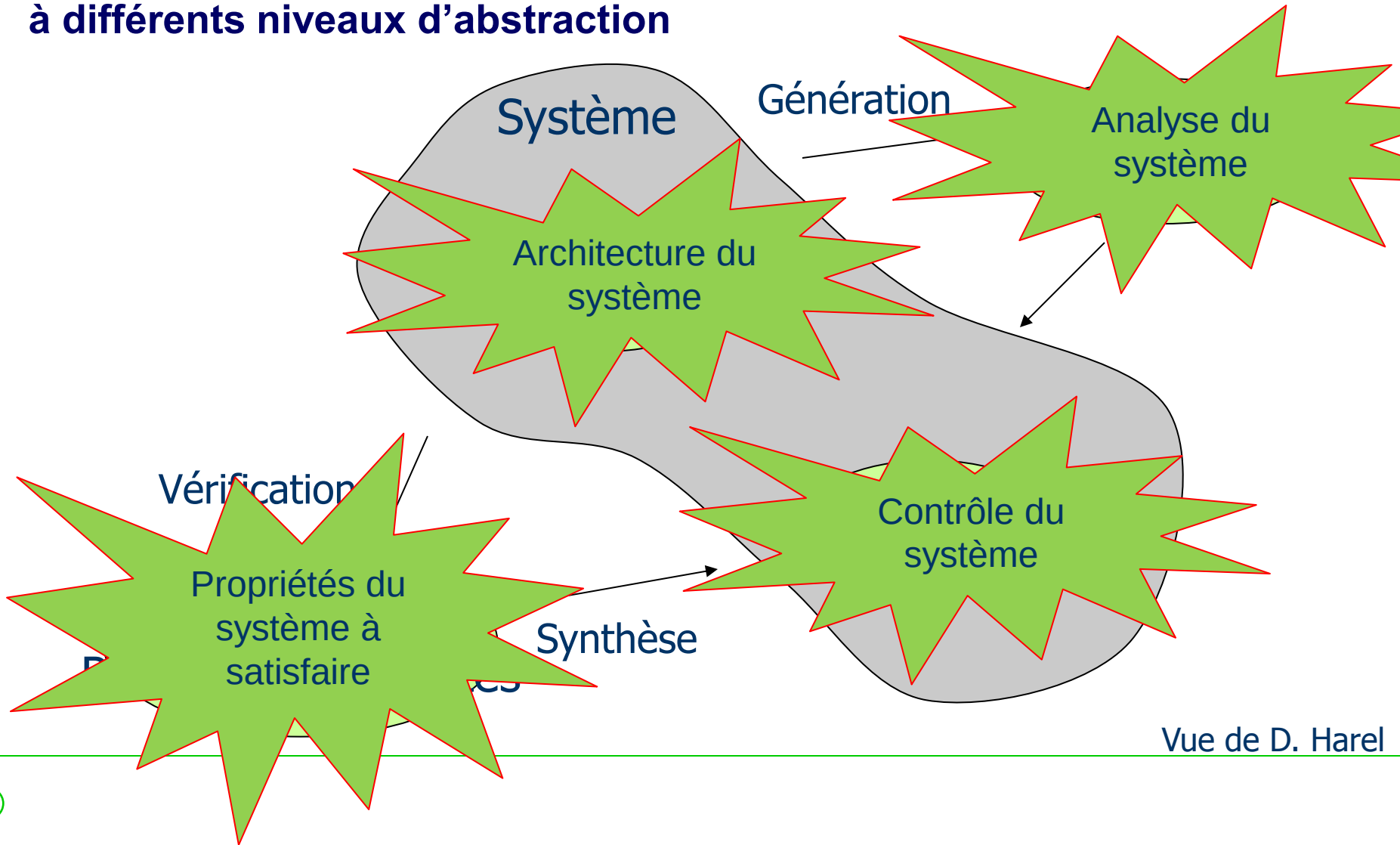
CreditCard **implies** High

Un modèle de système
avec les différentes préoccupations



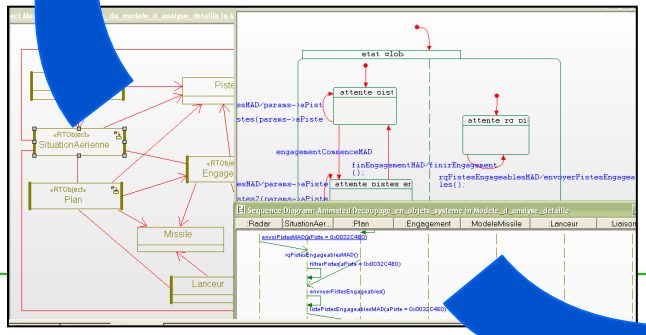
Vue de D. Harel

Des modèles système
à différents niveaux d'abstraction



Vue de D. Harel

From software entities to System of System



Abstraction and modelling

Systems and Models

- A Model by intention and by concern [Fav06,Mul10,P1471]
- A set of models to abstract a system

Using models and Modelling languages

- What modelling?
  - Concepts and relationships related to intention and concerns
- When modelling?
  - In the methodological steps
- Where modelling?
  - In the process development activities

Modelling the used of the models = Identify the role of the models :

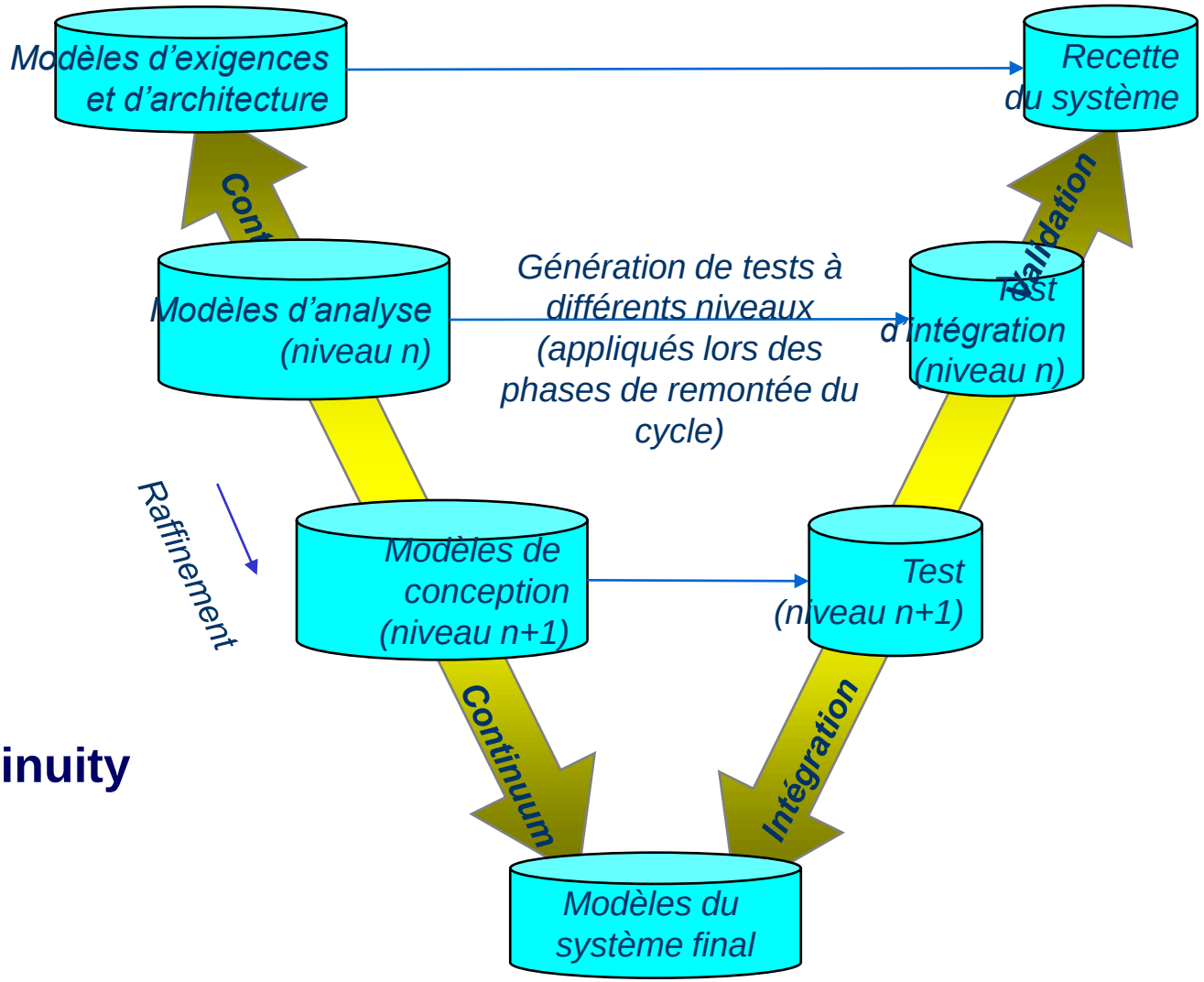
- Modelling the methods and processes, with BPMN, SPEM
- Modelling the relationships in the modelling space

From Software models to System models

Achieving a continuity in the development process

- Software heterogeneous = Modeled it and applied separation of concern
- Systems are naturally heterogeneous, complex, based on several concerns, several levels of abstraction
- Retrieving several problematic
  - Focusing on concepts and not on real artifacts
  - Defining architecture (functional, logical, physical, etc…)
  - Defining behaviors
    - Intern: state machine, data flow, control flow
    - External: interaction between entities
  - Many others concerns like deployment, user needs, requirements, integration scenario, etc…

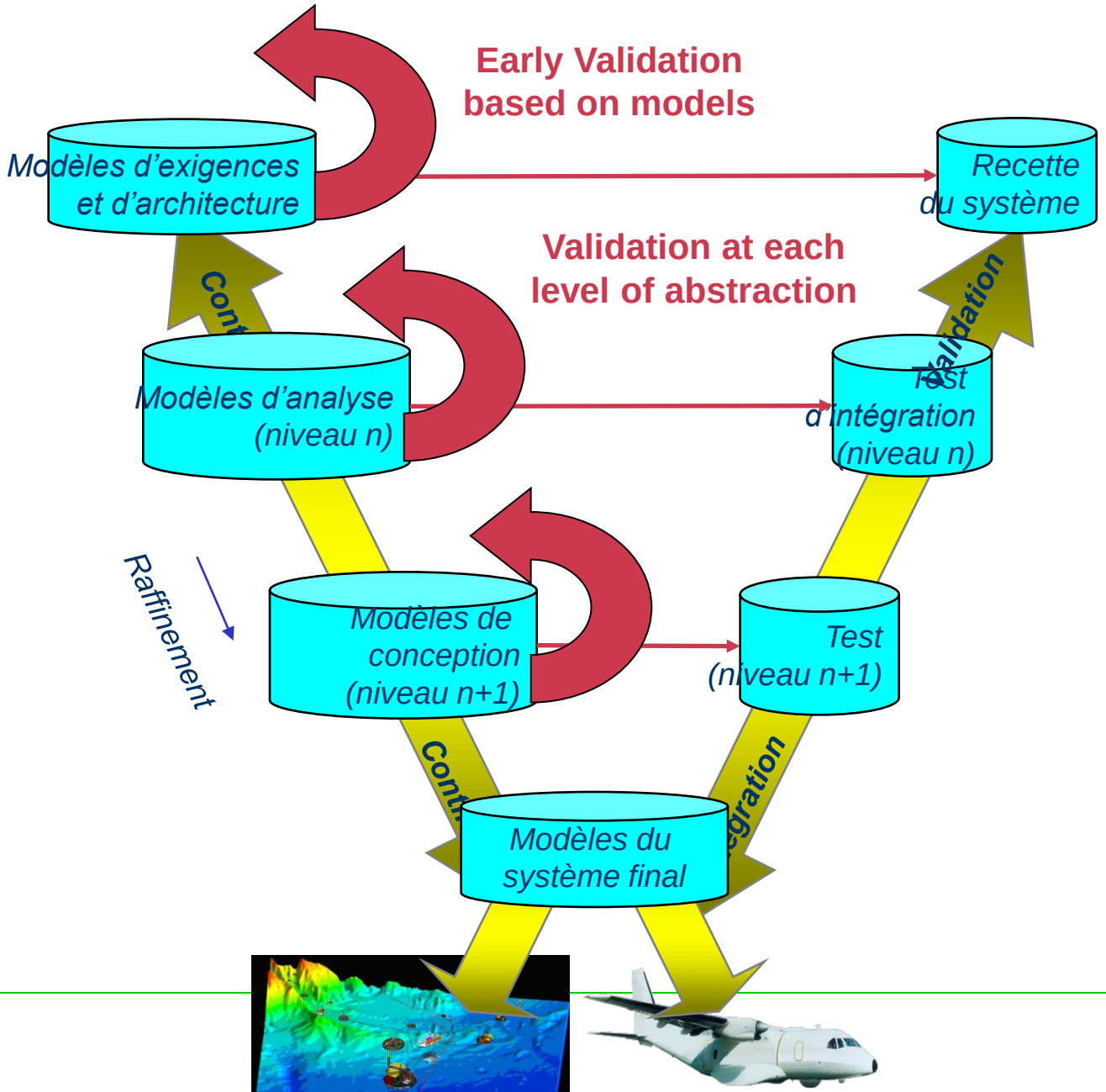
Modelling and processes



Creating a Continuity

in the process

Modelling and Processes



Defining a « generic » methodology

Thales R6 use case with the UML models associated

- The application : An anti-air system

Modélisation : Système Anti-Aérien

Plusieurs modèles exécutables

- Modèle d'analyse préliminaire
  - Objectif : Définir le contexte ou l'environnement du Système
- Modèle d'analyse détaillée au niveau système
  - Objectif : Modèle fonctionnel avec les objets systèmes qui portent les fonctionnalités du système
- Modèle d'architecture logique
  - Objectif : Etablir l'architecture des entités logiques (composants) organisant le système

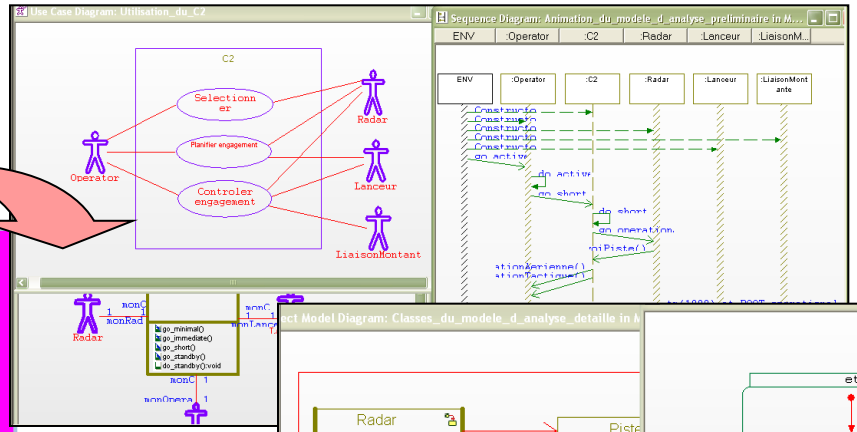
Model Driven Engineering

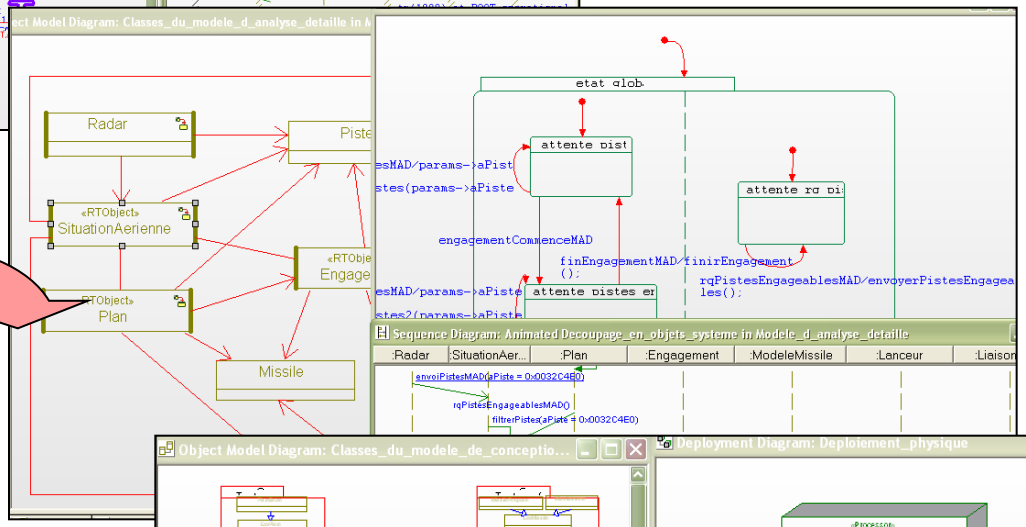
From UML set of models to …

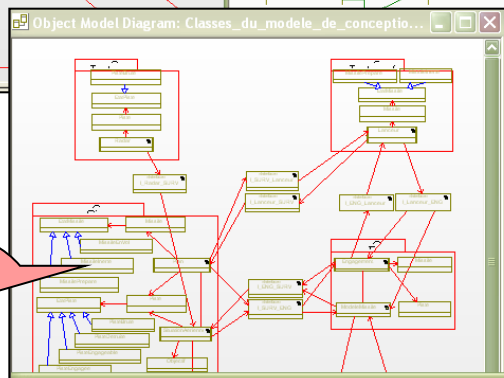
Context Model

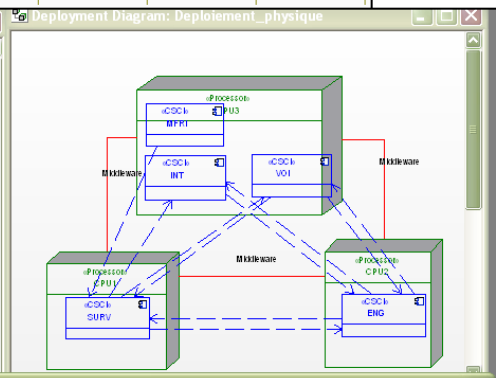
Business Model

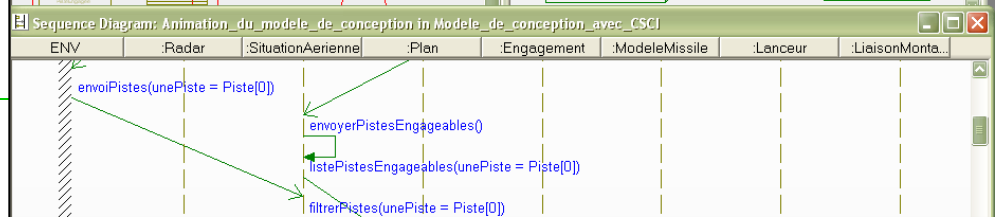
Design Model



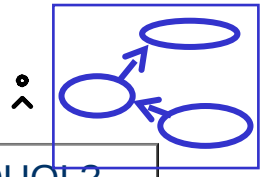






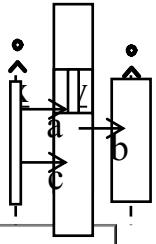


Modélisation Agile avec UML



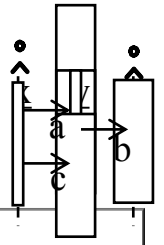
Le QUOI ?

Use Case



Trace

Diag Séquence



Analyse

Diag Séquence

Le comportement

Opérations et Statecharts

La structure

Diag Classe

Démo Rhapsody à suivre

Another example: MOPCOM SoC/SoPC project

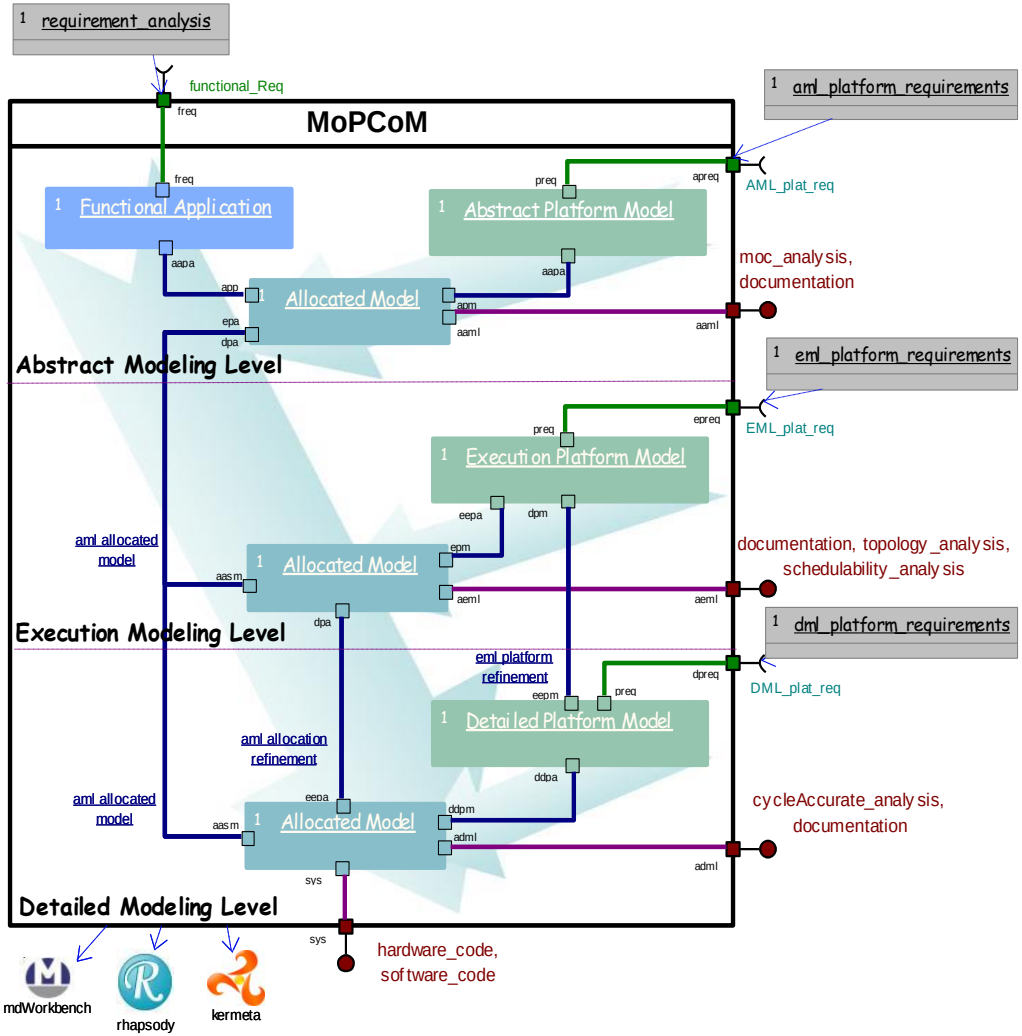
MOPCOM SoC/SoPC ANR project 2007-2010.

3 abstraction levels:

- Abstract Modeling Level: Explicitation on virtual machine
- Executing Modeling Level: Performances analysis on abstract machine
- Detailed Modeling Level : Cycle accurate on detailed platform.

Each level = a set of models.

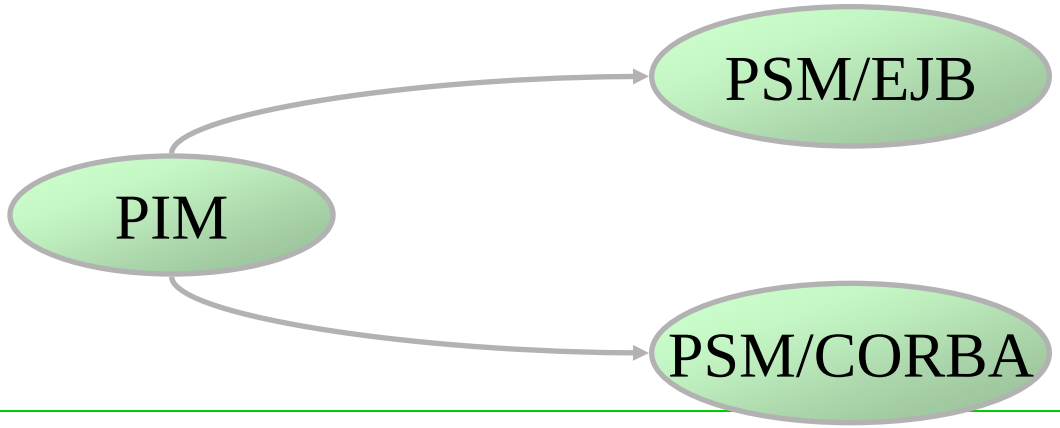
- Business model
- Platform model
- Allocation model
- Analysis model



Démo MDWorkbench à suivre

MDA - Model Driven Architecture – Historic approach

- Platform Independent Modeling
  - Model classification.
  - Platform = service software layer.
- Classes of models
  - PIM = Platform Independent Model
  - PSM = Platform Specific Model

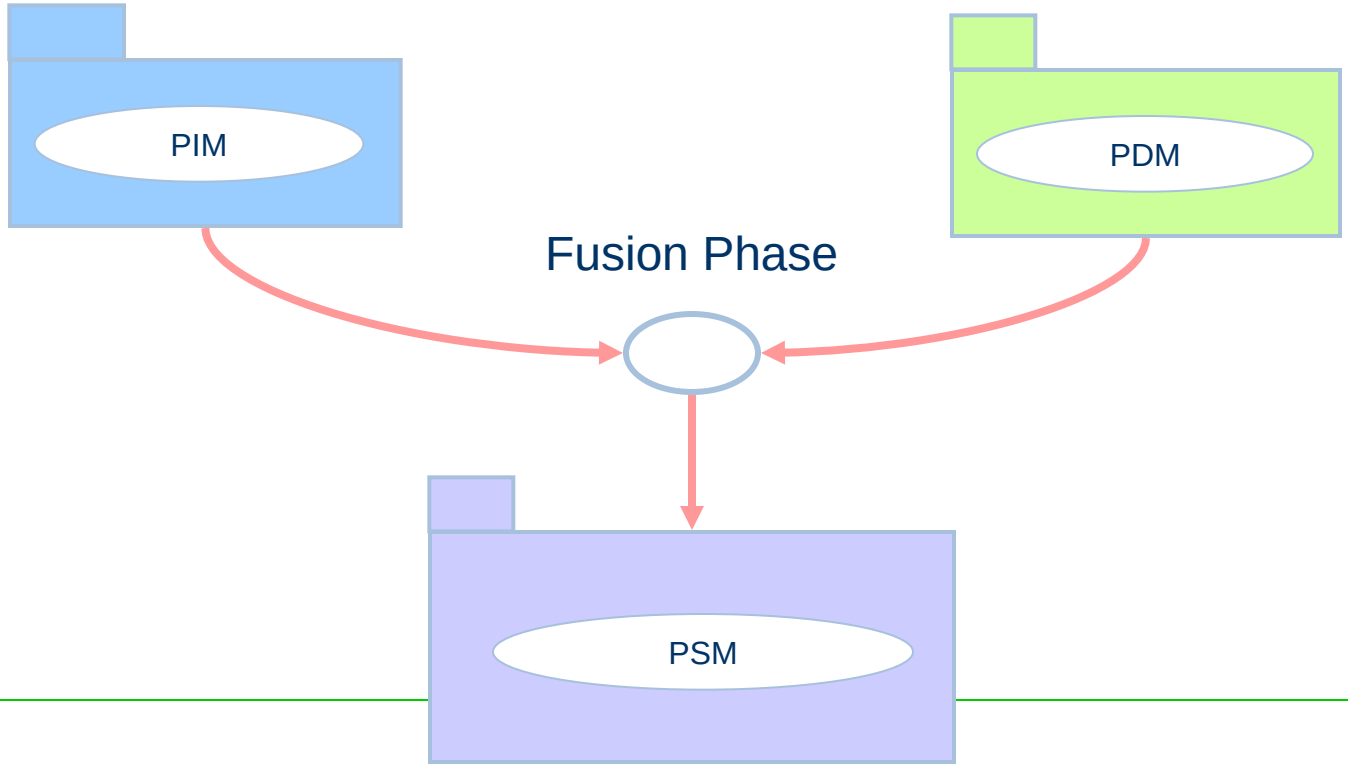


Or EJB

Use of known profile for mapping, CORBA

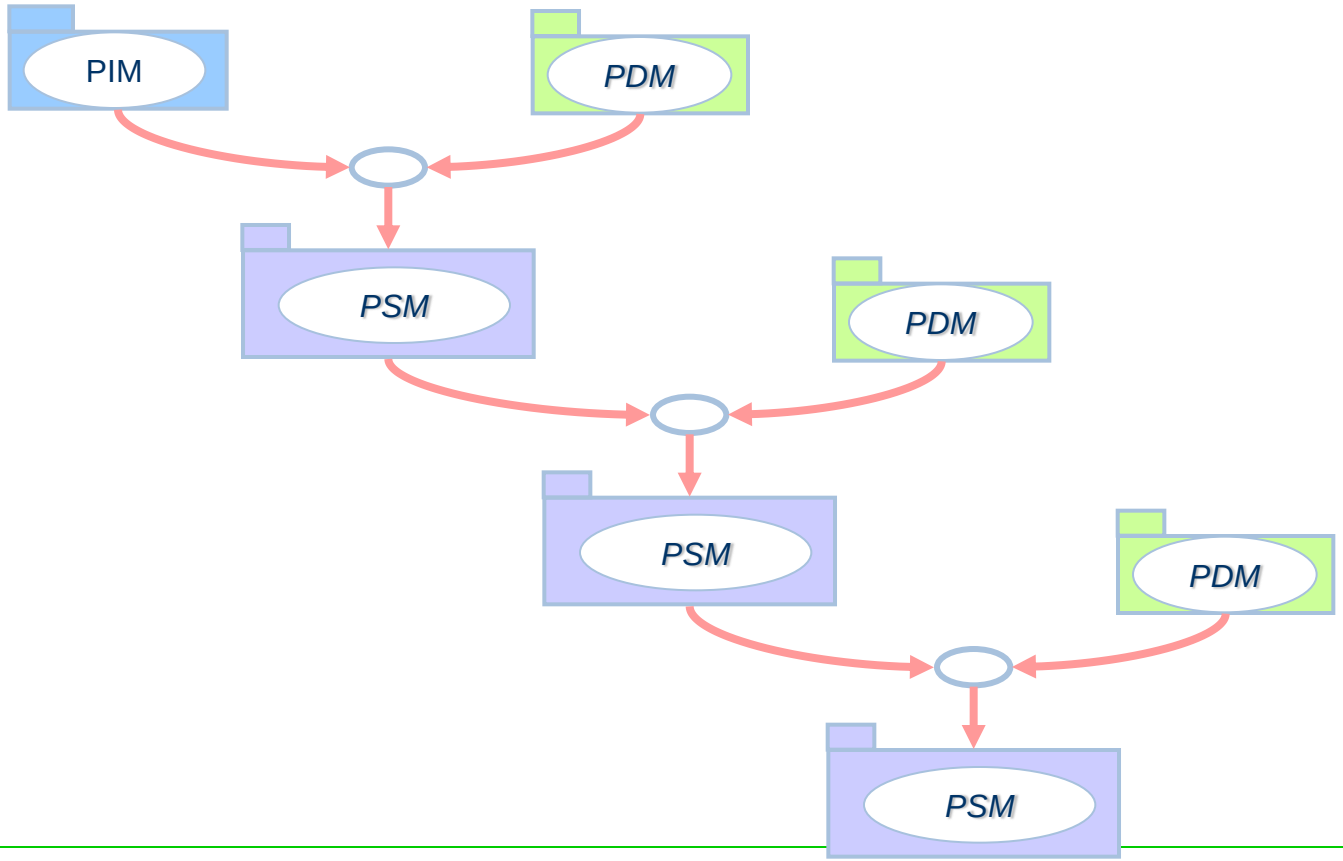
MDA pattern

- Revisited development cycle
  - Y based
  - PIM, PSM and PDM = Platform Description Model



MDA process

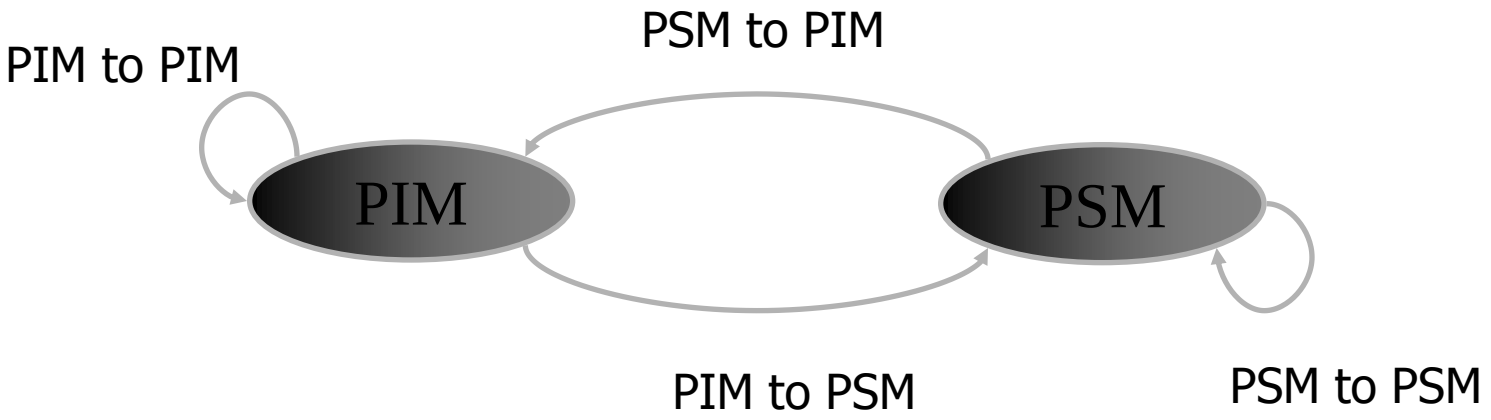
- PIM, PSM and PDM are relative = Role Notion



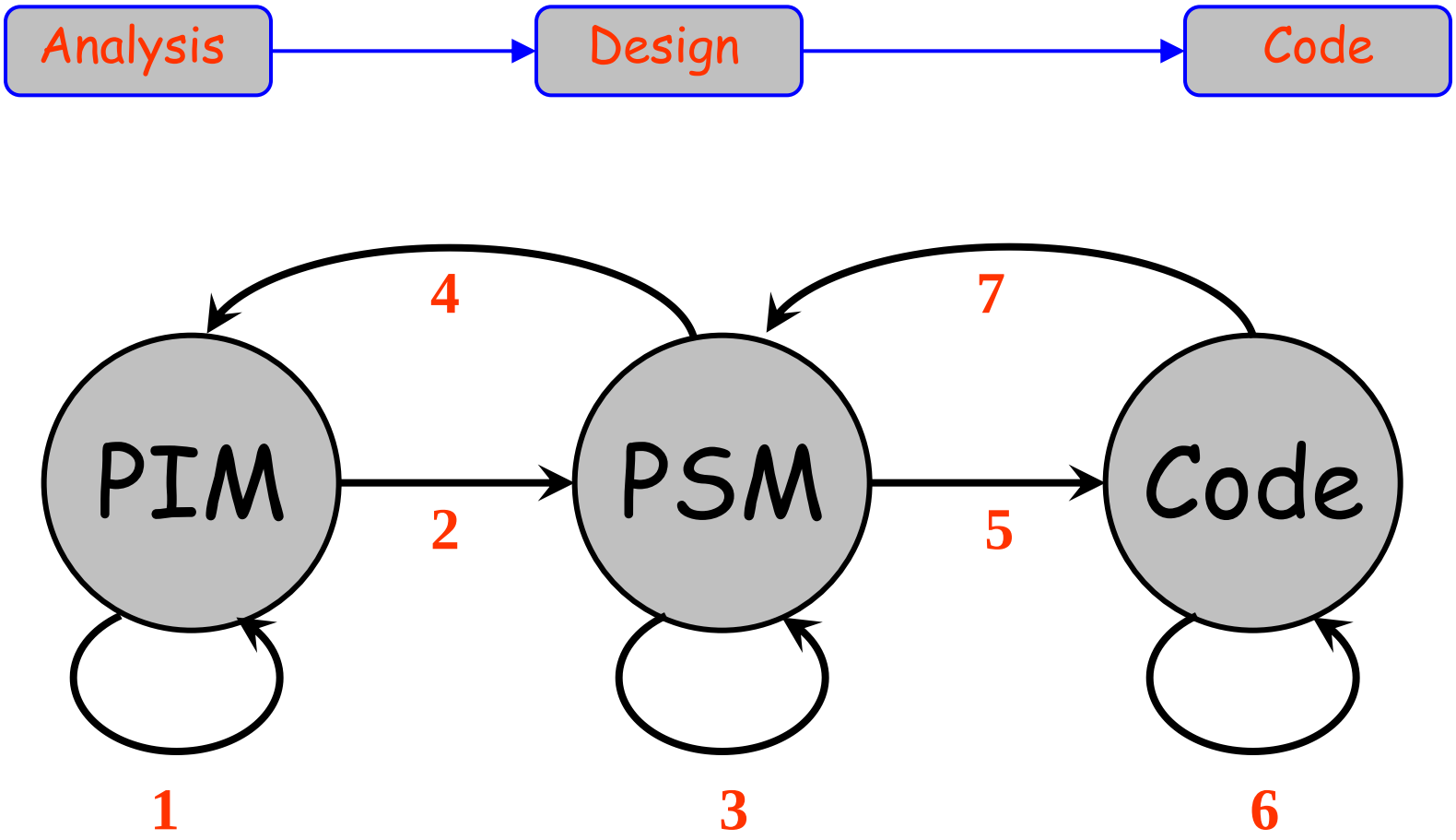
Transformations in the MDA context

Model transforming linked to different types.

- Extension of the mapping notion to the transforming notion.
  - Classification of the transformations.



Development Cycle



Operation Classes

Transformations and meta-model :

- **In the scope of a meta-model**
  - Refactoring
  - Refinement
- **In the subset of a meta-model.**
  - Common parts to several meta-models.
- **Between two different meta-models**
  - Generalization de N to N.
  - From a formalism to an other

Transformations Operations

Simple operations :

- Creation, Modification, Navigation, Suppression, Display
- Serialization
- Extension, Reduction
- Measures of elements
- Code Generation, Traceability

More general operations :

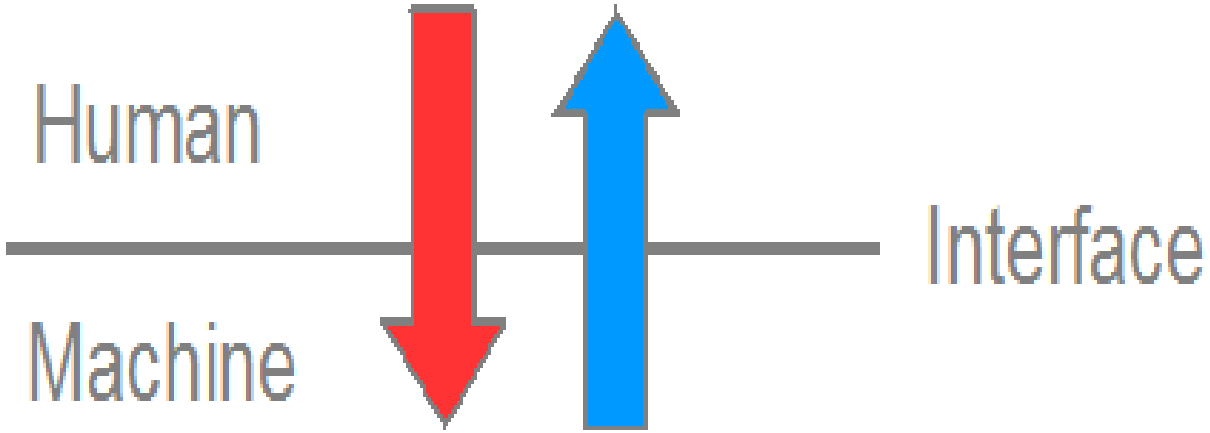
- Comparison
- Veawing
- Fusion
- Alignment
- Reversible operation = Tracks production
- Abstraction

Un exemple : Interface Homme - Système

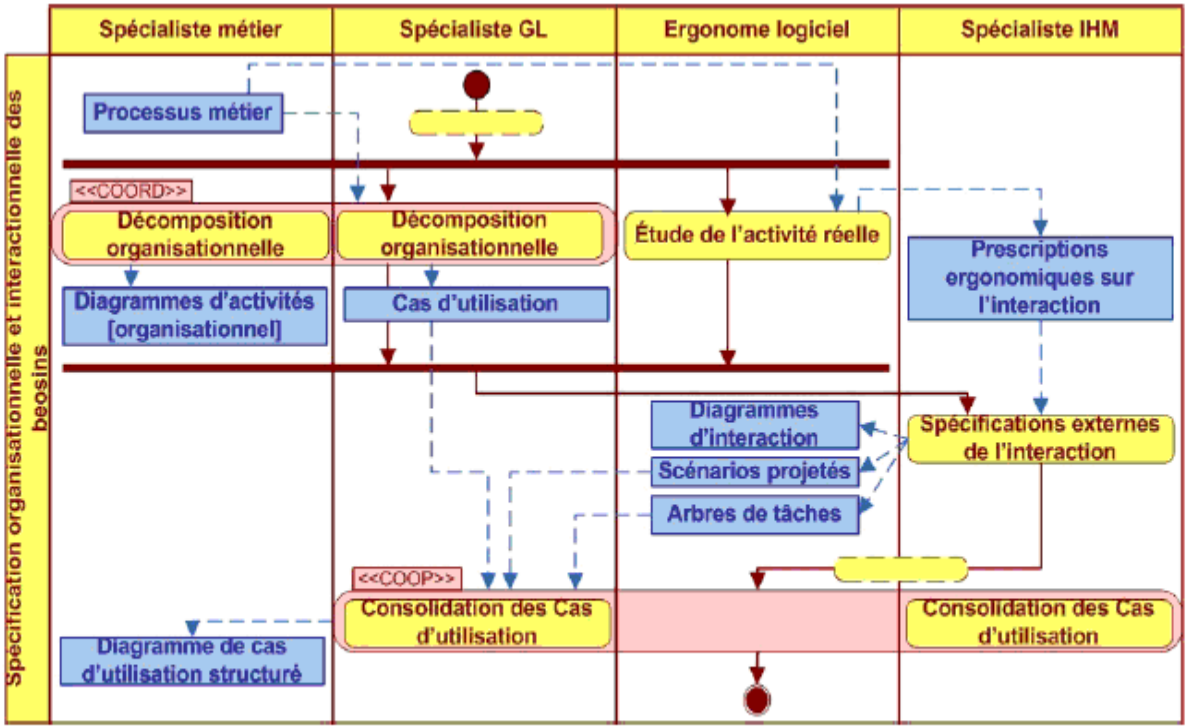
Projet MEDUSA 2011-2014 (Telecom Br, ENAC, Thales, Sodius, Ensta Br)

Modélisation pour la conception d'IHM orienté usages.

Objectif : Guider la conception des IHMs du niveau système jusqu'à la réalisation



Methodologie Symphonie pour IHM



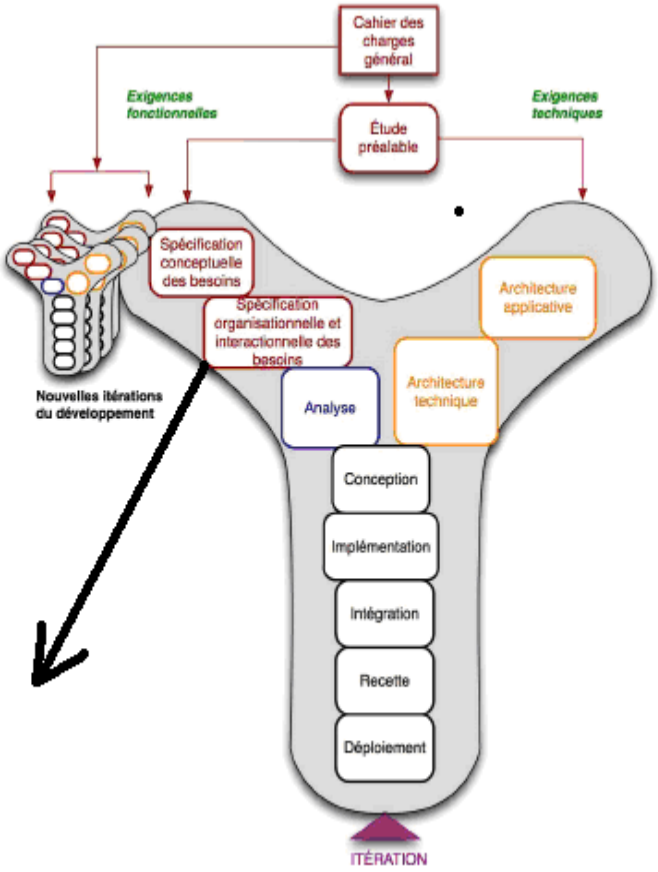


FIGURE 59. ACTIVITES DE SPECIFICATION ORGANISATIONNNELLE ET INTERACTIONNELLE DES BESOINS DE LA METHODE SYMPHONY ETENDUE [JURAS ET AL., 2006]

Méthodologie Symphonie pour IHM

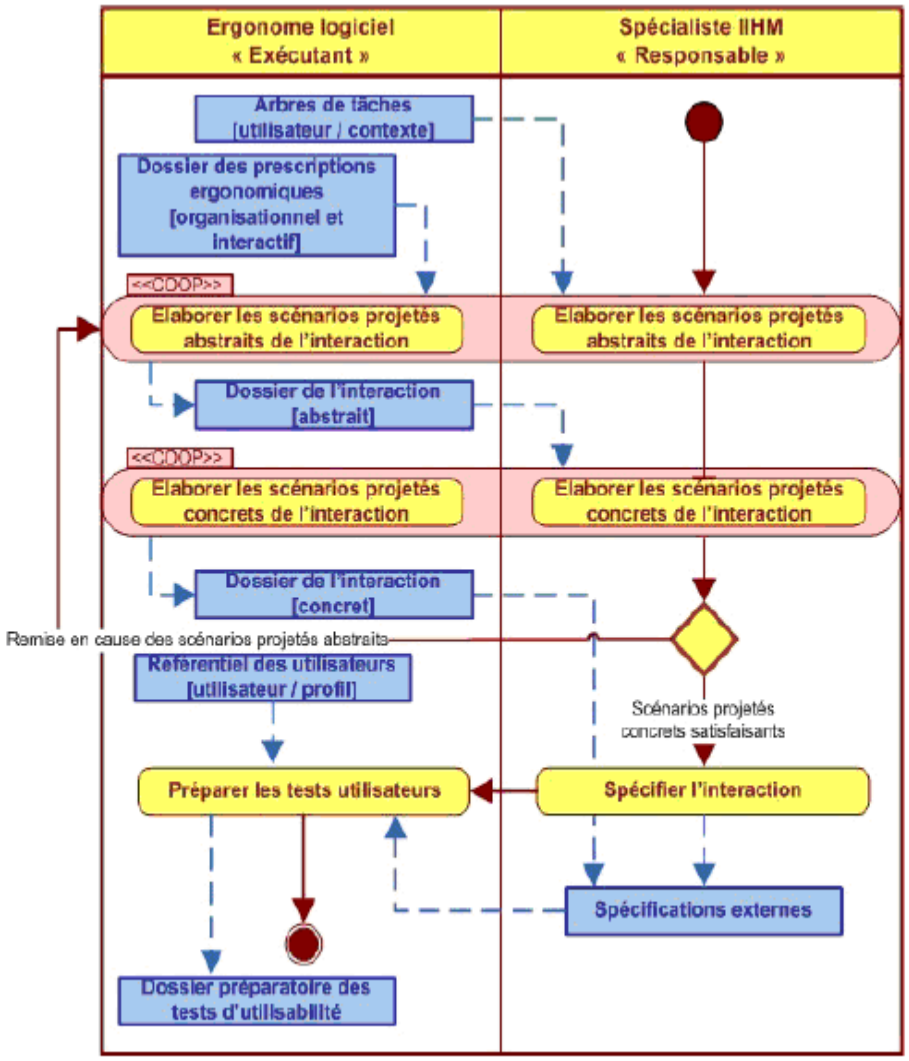


FIGURE 60. ACTIVITE DE SPECIFICATION EXTERNE DE L'INTERACTION DE LA METHODE SYMPHONY ETENDUE [JURAS ET AL., 2006]

Process activities

- Platform Metamodels (in Y) are based on Domain Concepts (IHM)
- From standard like UsxML (Concrete User Interface Metamodel)
- Process rupture between models and recommendation reports

IHM and Modelling

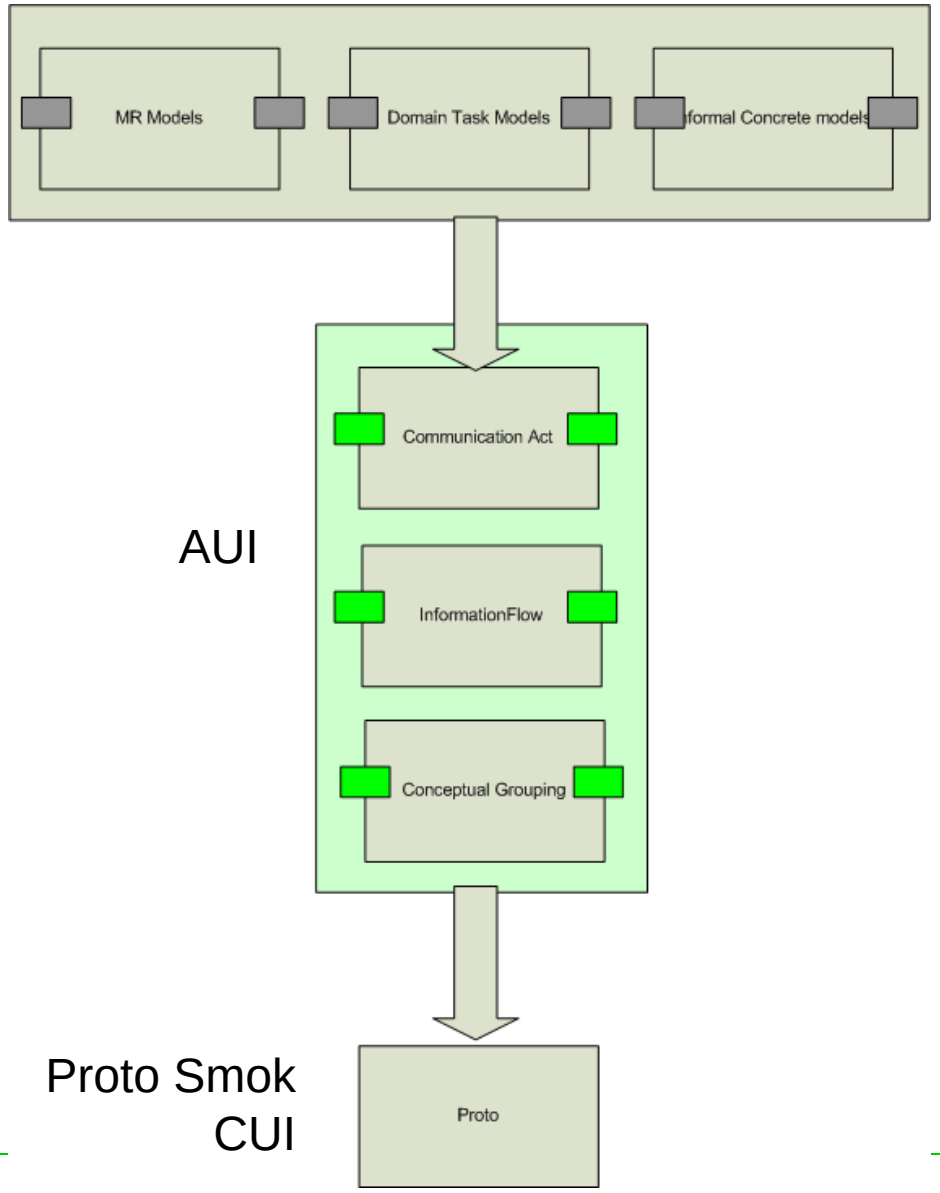
Process activity and metamodel identification

- Platform Metamodels (in Y) are based on Domain Concepts (IHM)
- Sub-set of standard metamodel like UsxML
- Dedicated metamodels to target dedicated concern
- Several abstract levels in metamodels

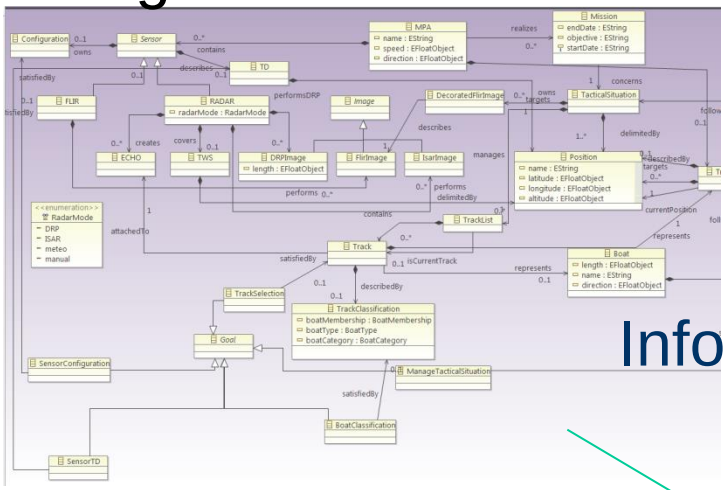
Process activity (Role) and metamodel constraints

- Methodological constraints on metamodels
- Properties included in the metamodels
- Easier to obtain metrics on used concepts

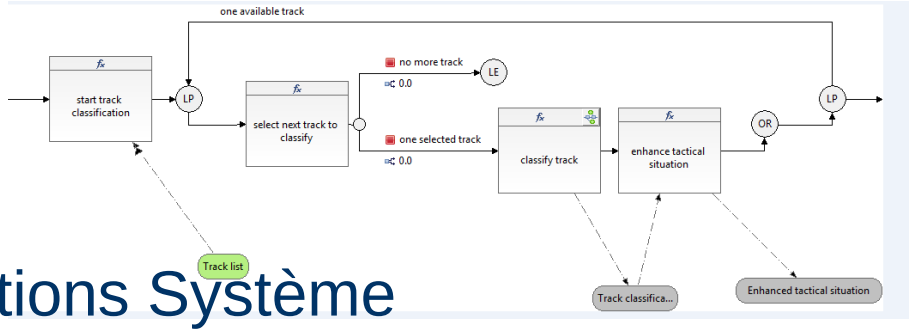
IHM and Modelling Process



Ontologie et Modèle de RM



Modèle de tâche



Informations Système

Génération semi –automatique

AUI

Groupement Data

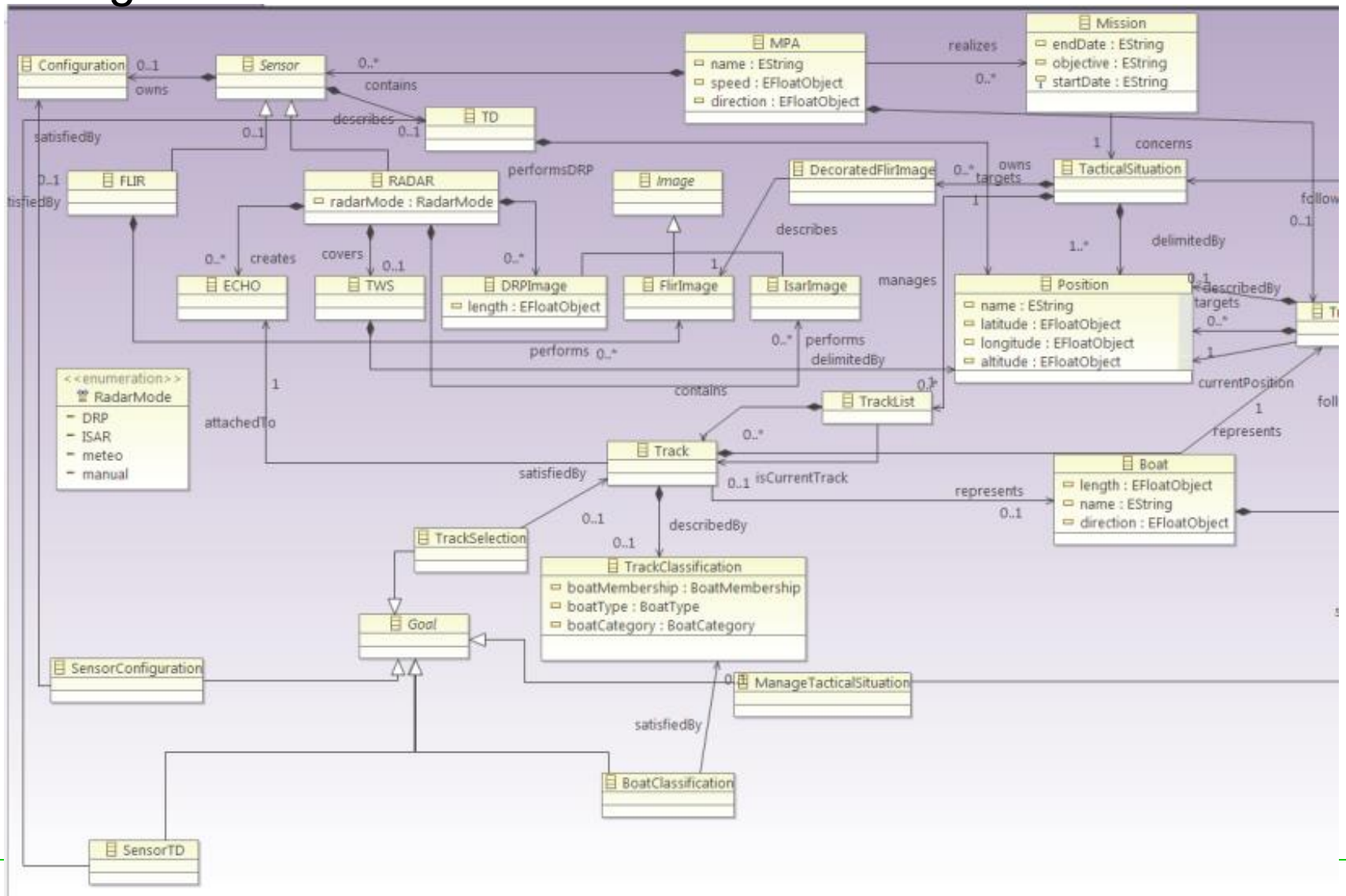
Information flow

Choice

CUI

CUI conçue à partir de l'AUI qui est une spécification

Ontologie et Modèle de RM



Contexte : Metamodelisation et IHS

- Indépendamment et après l'analyse système
- Systèmes réactif
- Aspect ontologique du domaine mais avec une finalité langage de modélisation

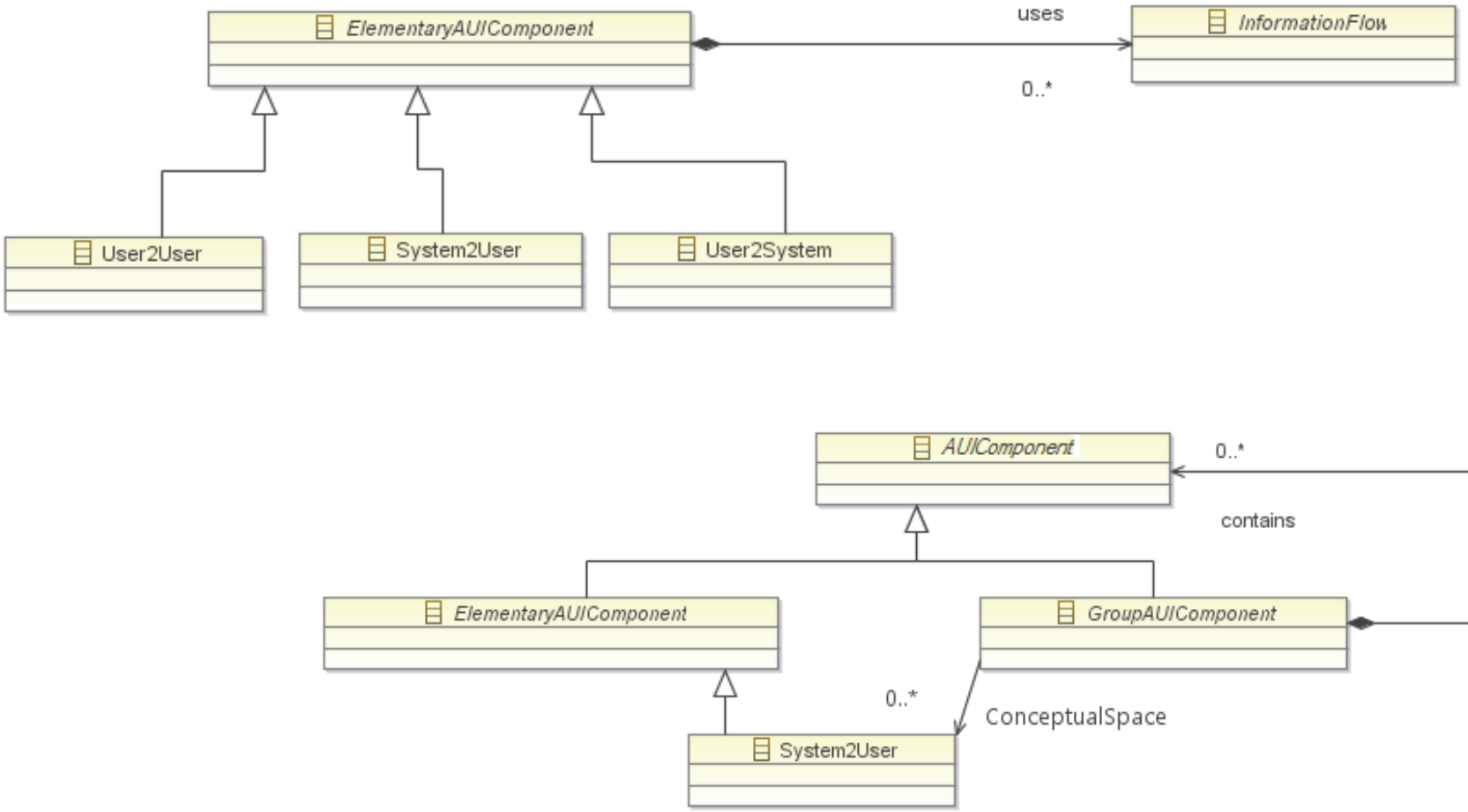
Besoins AUI

- Analyse du modèle AUI
- Modèle structurel
- Chercher à identifier le contrôle associé à l'IHS
  - Indépendant des modalités
  - Notion d'activation/desactivation
- Flux d'informations et contrôle

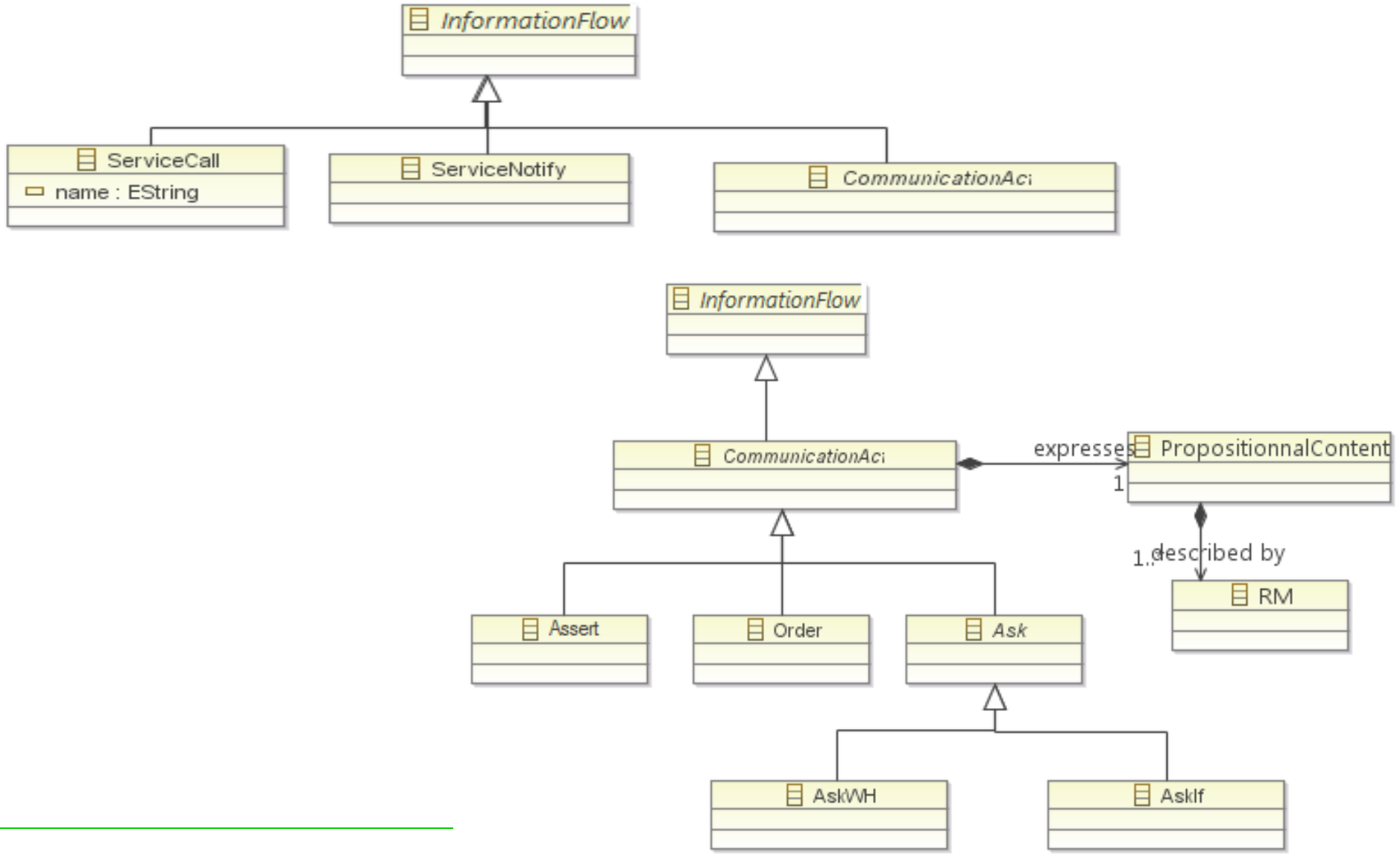
Metamodèle AUI

- Une préoccupation formalisée (lien système – CUI)
- Un aspect ontologique du domaine avec un comportement local
- Spécification du contrôle des composants

Composants élémentaires



Information Flow



Différents modèles conceptuels de données

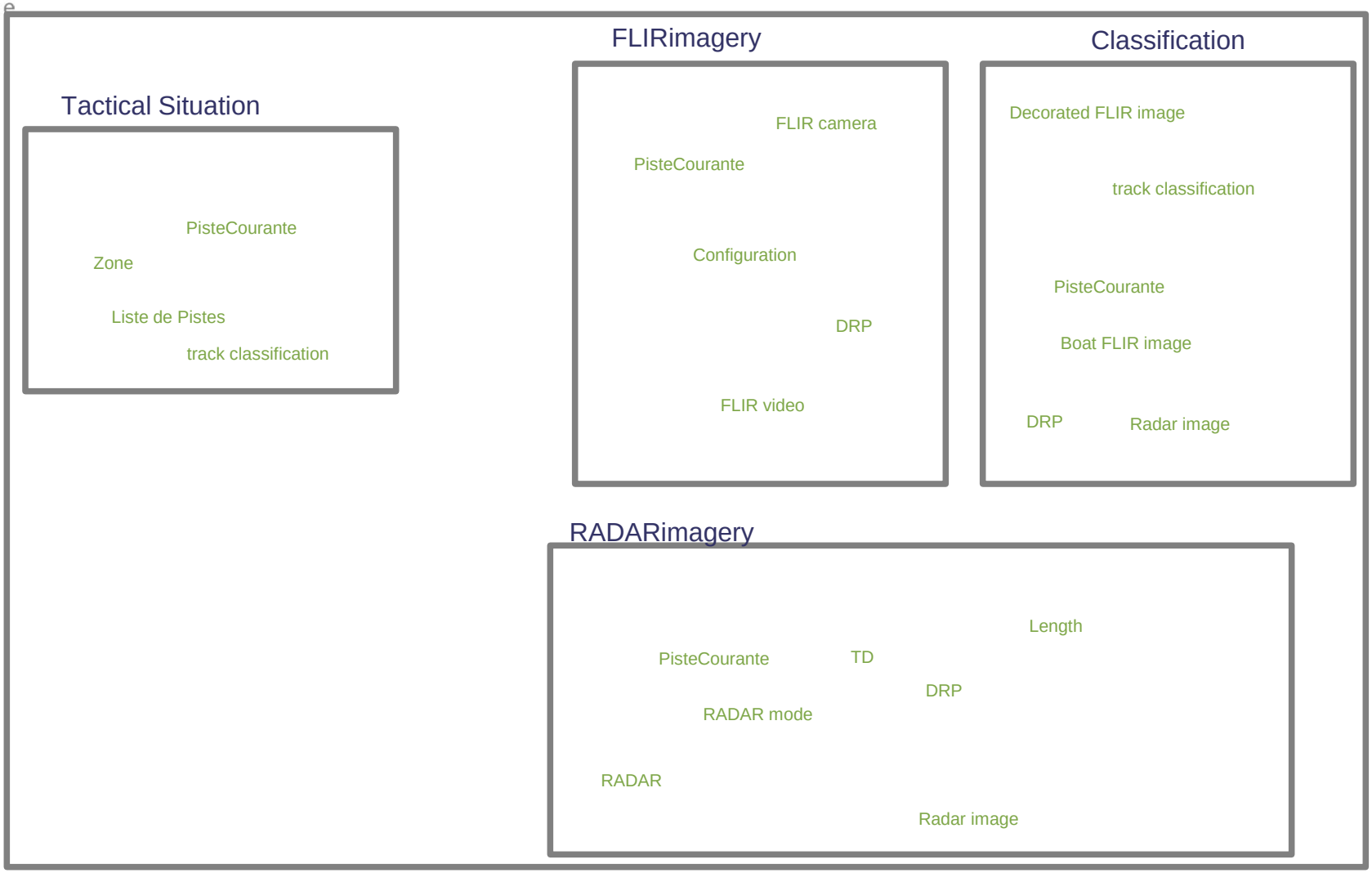
Modèle conceptuel de données au niveau système

Modèle cognitif opérateur avec une représentation mentale de la situation

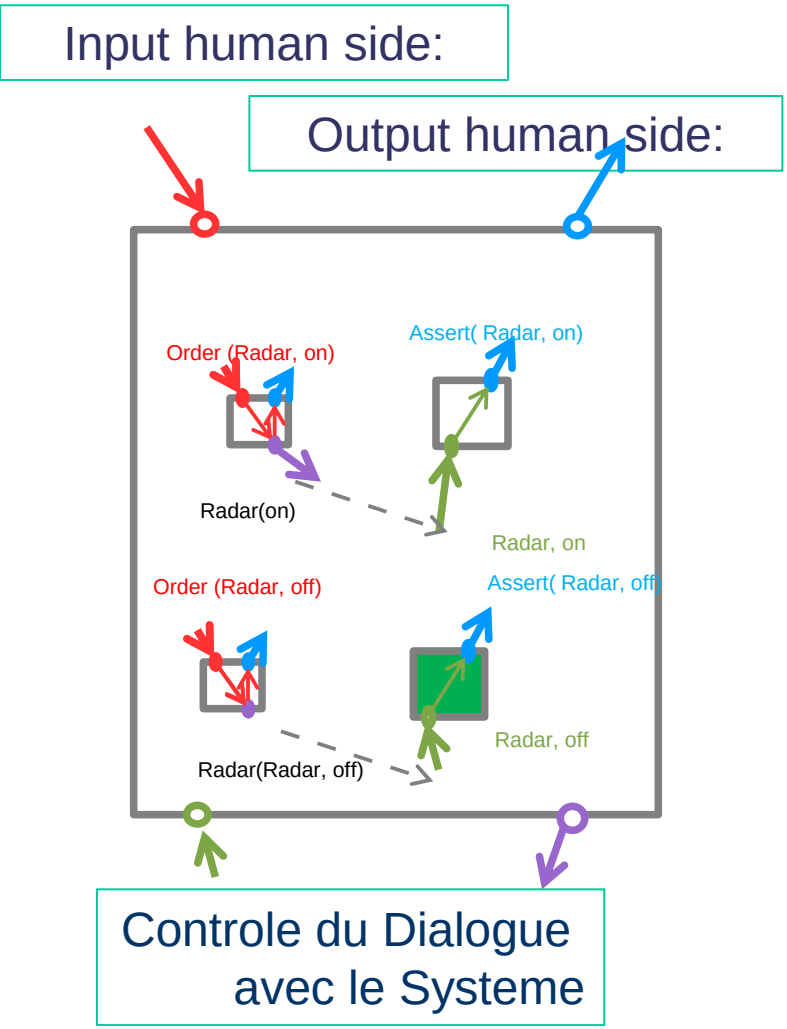
- aspect conceptuel
- aspect dynamique

Modèle conceptuel de données de l'IHM

- Cycle de vie
- Représentation : visuelle graphique, textuelle, sonore, … modale
- Codage/numérisation pour stockage sur un support, transmission
- Nature : discrète, continue, globale, structurée vs élémentaire


Tactical Situation
PisteCourante
Zone
Liste de Pistes
track classification
FLIRimagery
FLIR camera
PisteCourante
Configuration
DRP
FLIR video
Classification
Decorated FLIR image
track classification
PisteCourante
Boat FLIR image
DRP
Radar image
RADARimagery
Length
PisteCourante
TD
DRP
RADAR mode
RADAR
Radar image

Expérimentation du langage AUI



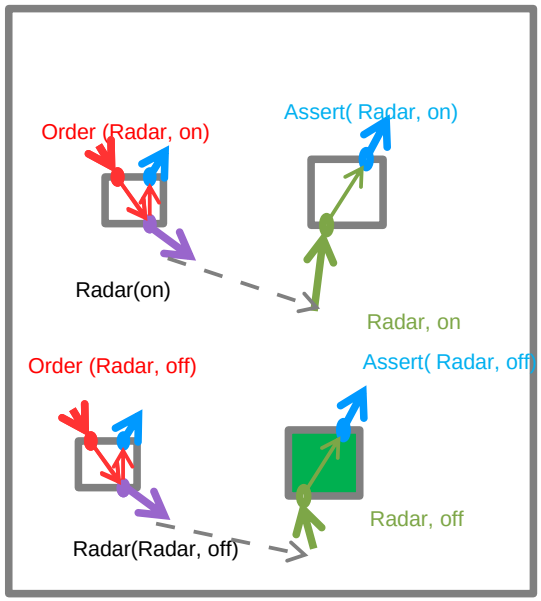
Identification du contrôle AUI

Expérimentation :

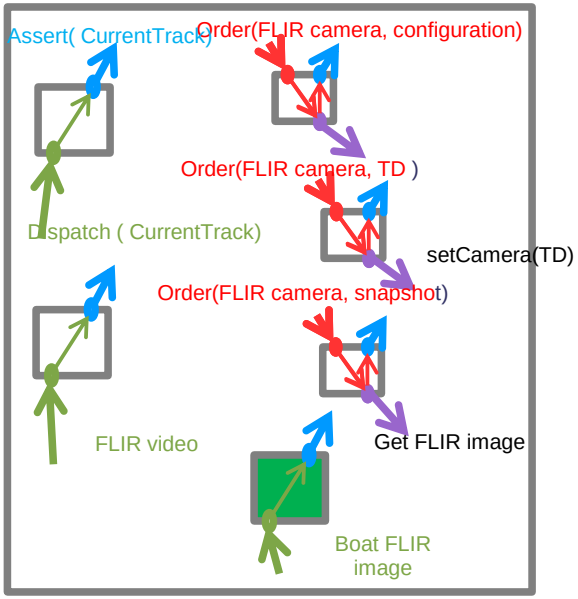
- 1 groupement
- composants élémentaires User2System
- composants élémentaires System2User

2 exemples simples d'abord

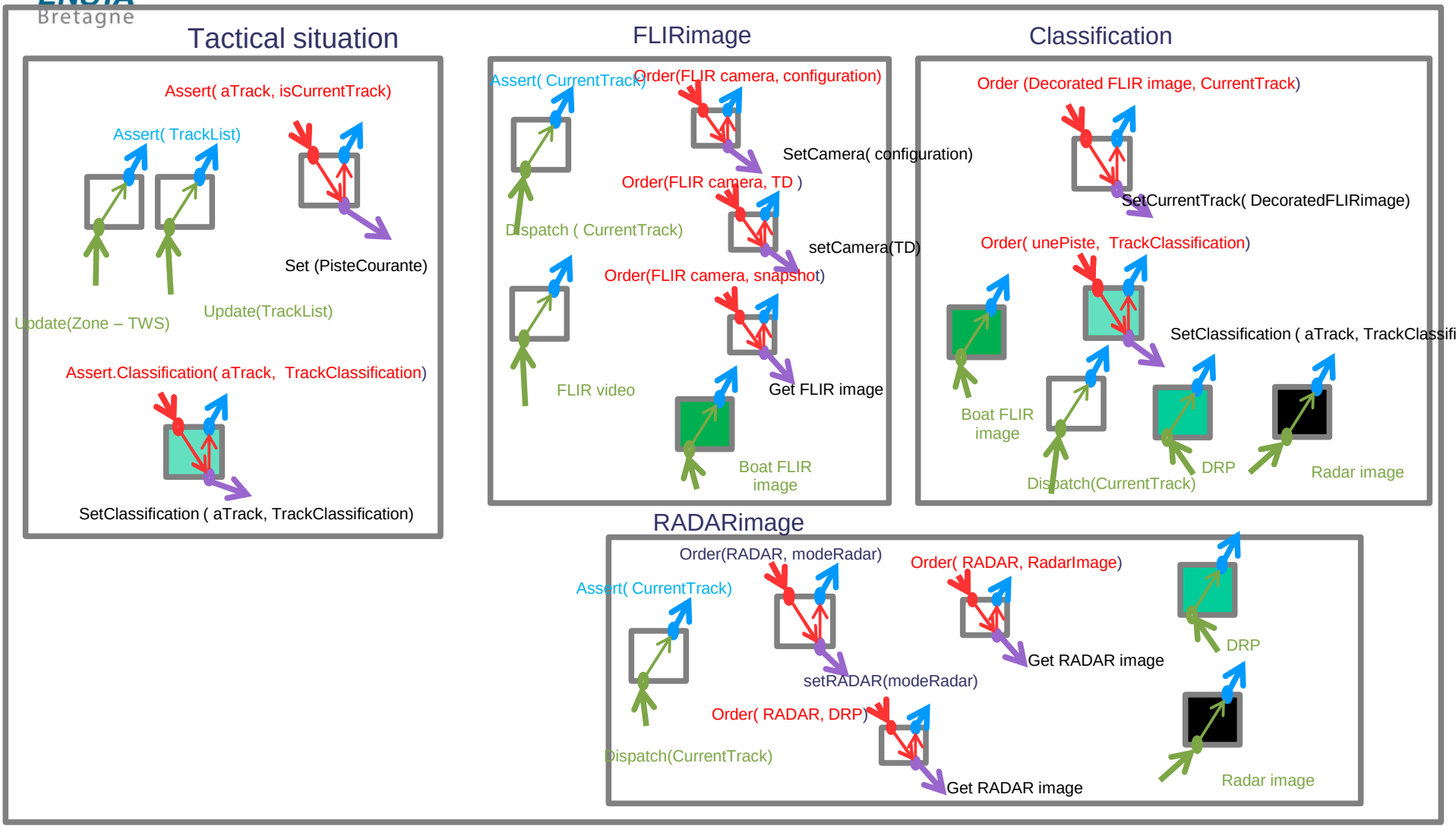
RadarControl



FLIRimage



Décomposition en composants élémentaires

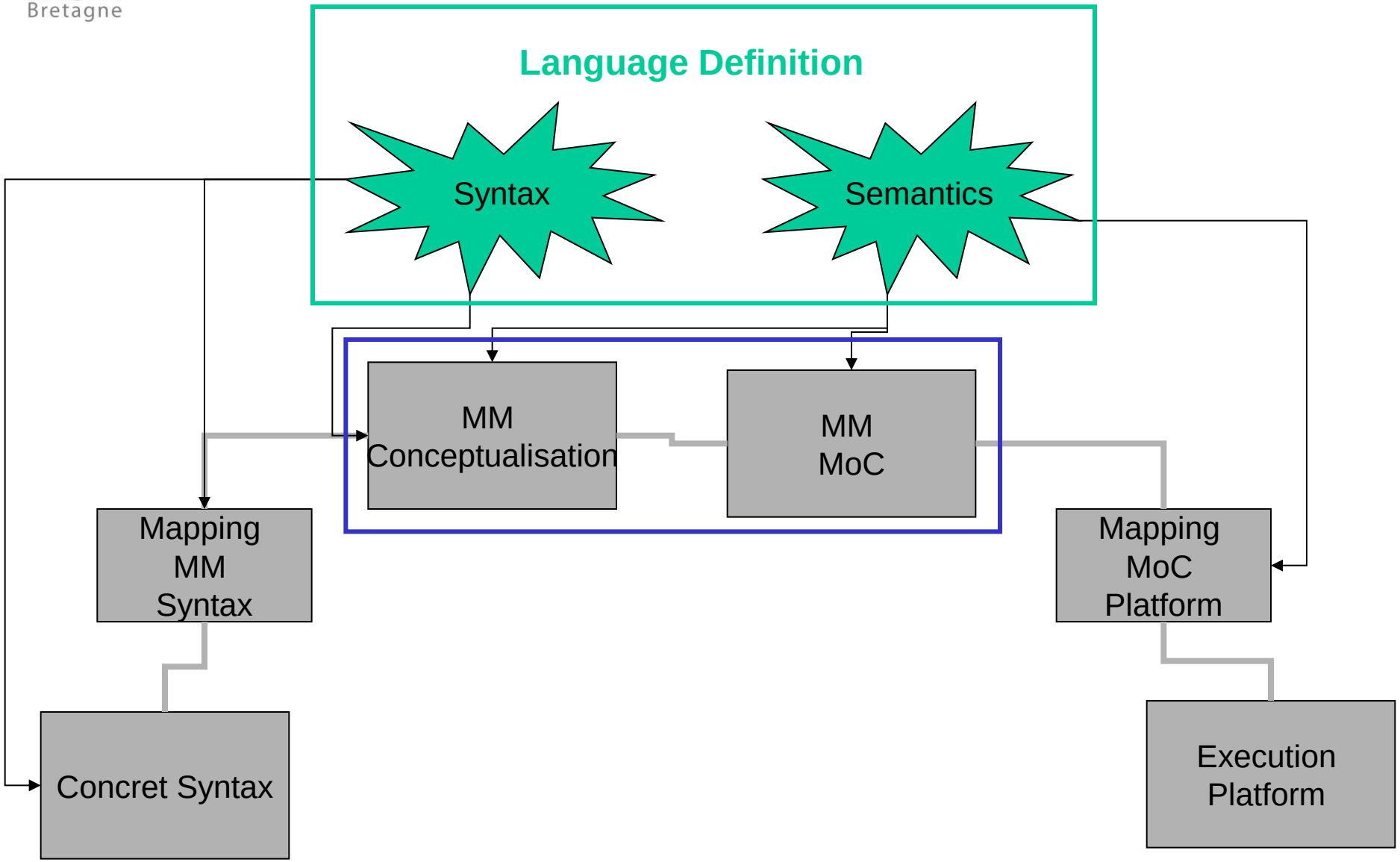


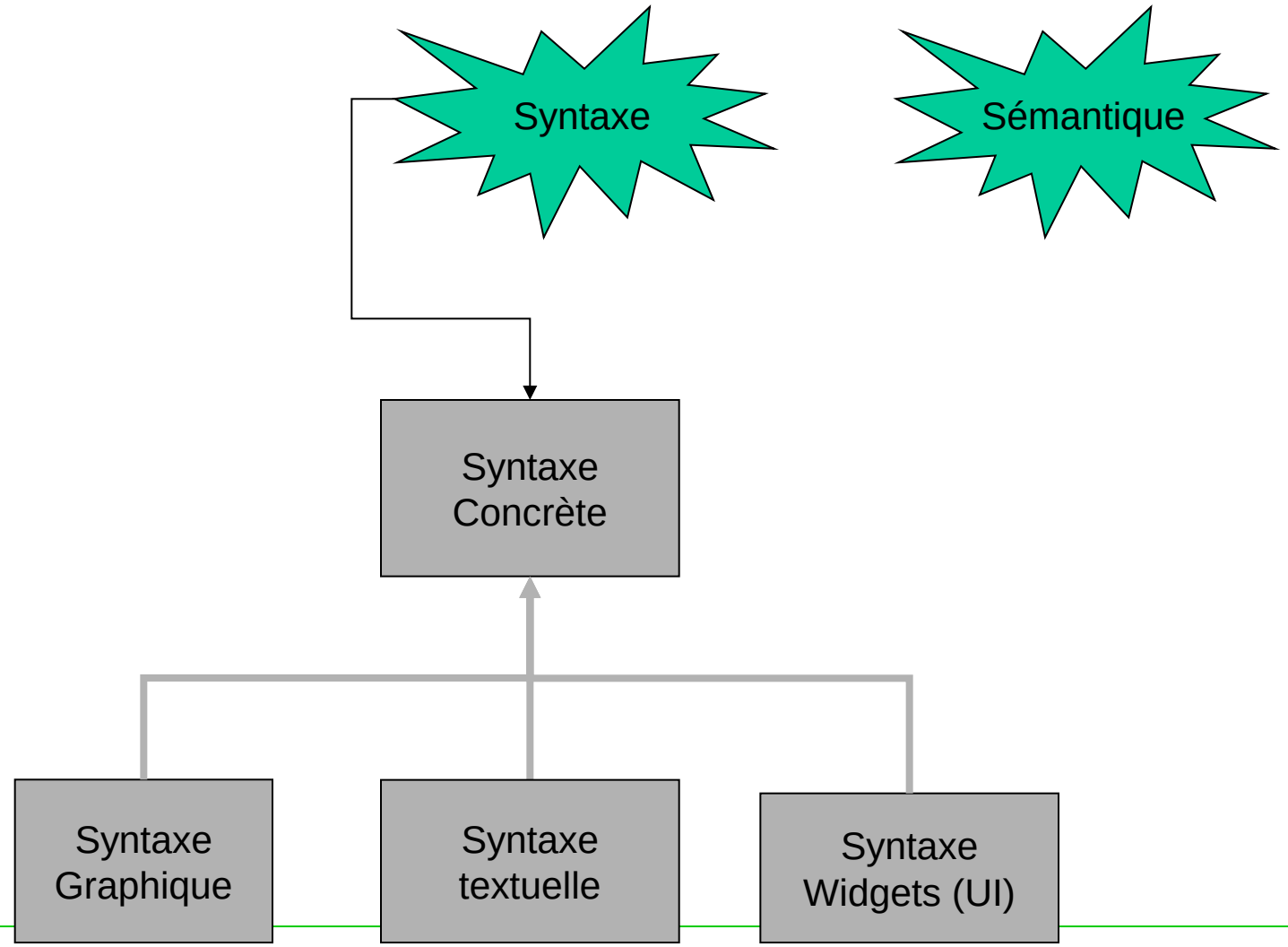
Conclusion

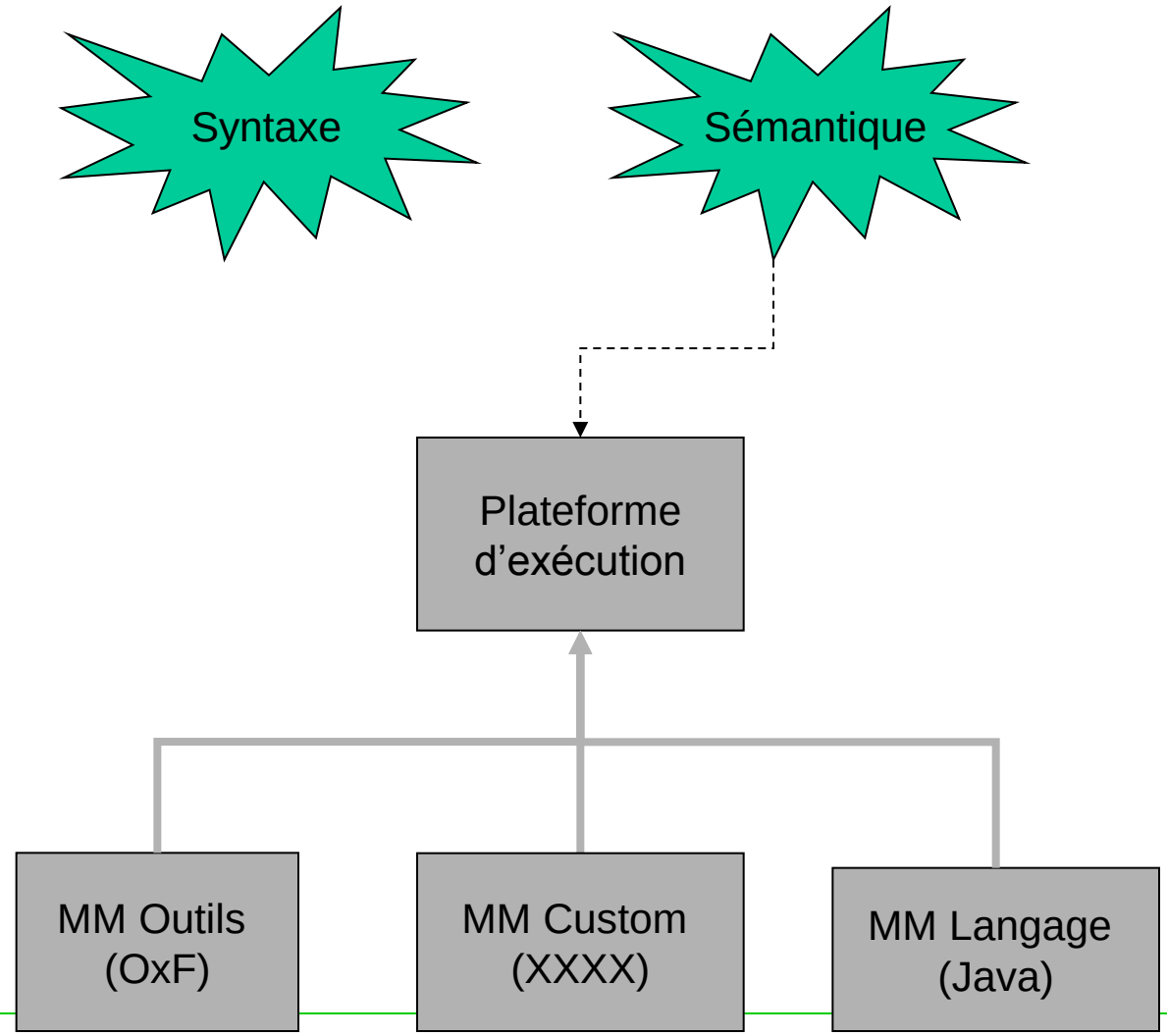
- Un metamodèle existe pour l'AUI
- Aspect ontologique fort
- Aspect structurel sur des regroupements ou composants
- Réceptacle possible de transformation de modèle possible
- Exécution toujours possible mais quelle est sens de cette exécution ?

ENSTA Bretagne

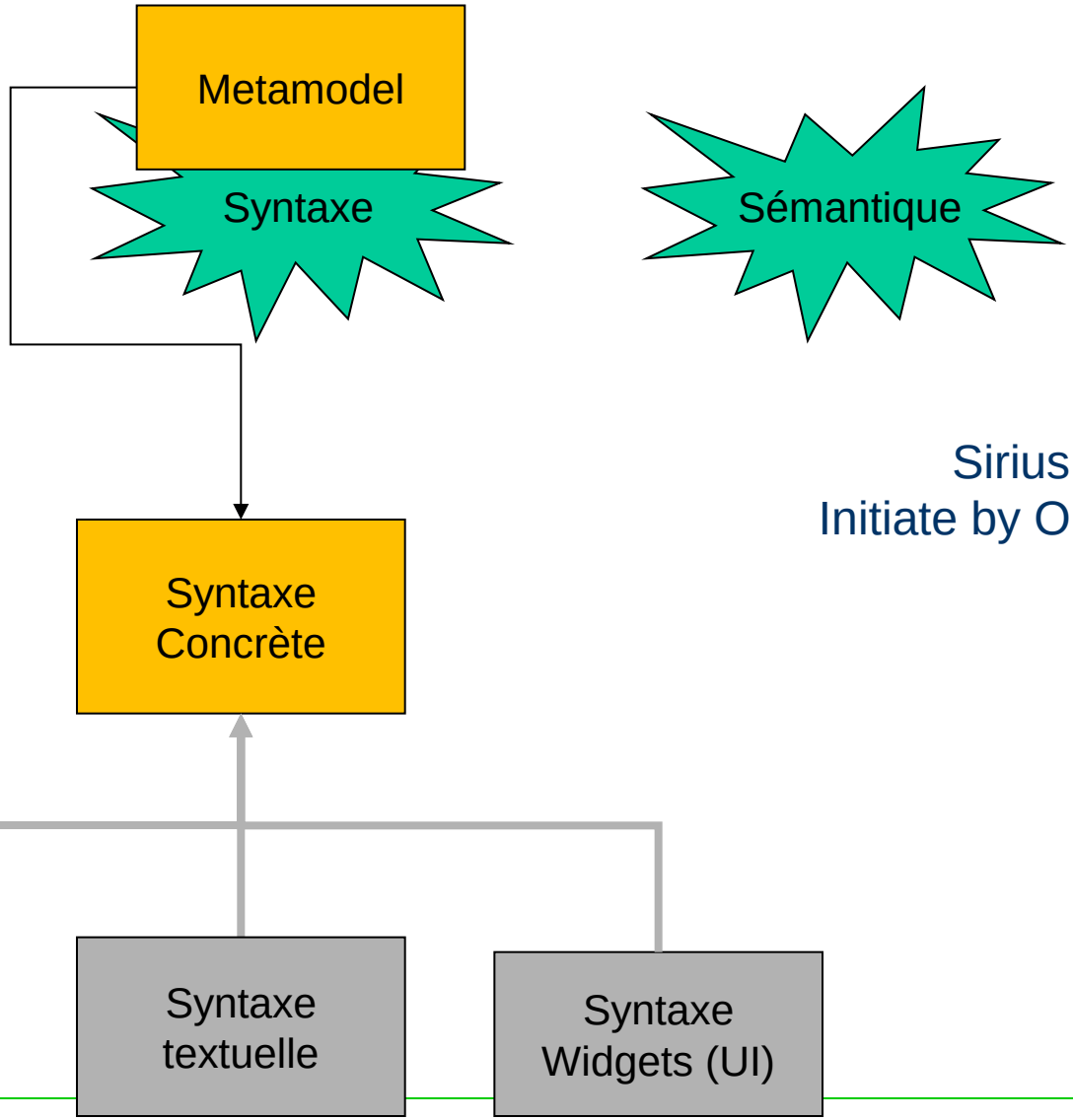
Metamodel, Syntax, Operationnal Semantics for Modeling Language


Language Definition
Syntax
Semantics
MM
Conceptualisation
MM
MoC
Mapping
MM
Syntax
Mapping
MoC
Platform
Concret Syntax
Execution
Platform


Syntaxe
Sémantique
Syntaxe Concrète
Syntaxe Graphique
Syntaxe textuelle
Syntaxe Widgets (UI)


Syntaxe
Sémantique
Plateforme d'exécution
MM Outils (OxF)
MM Custom (XXXX)
MM Langage (Java)

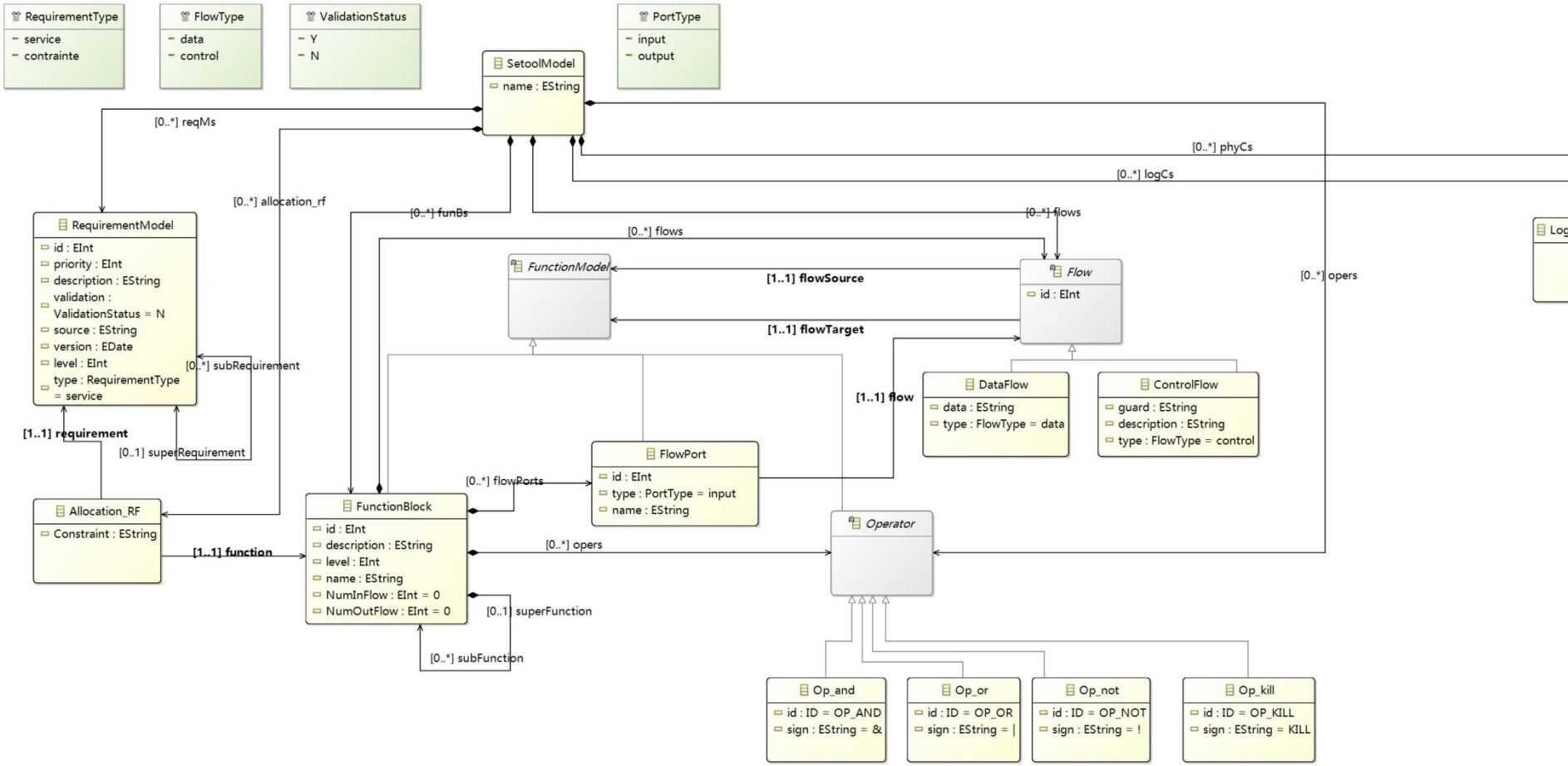
Sirius Context



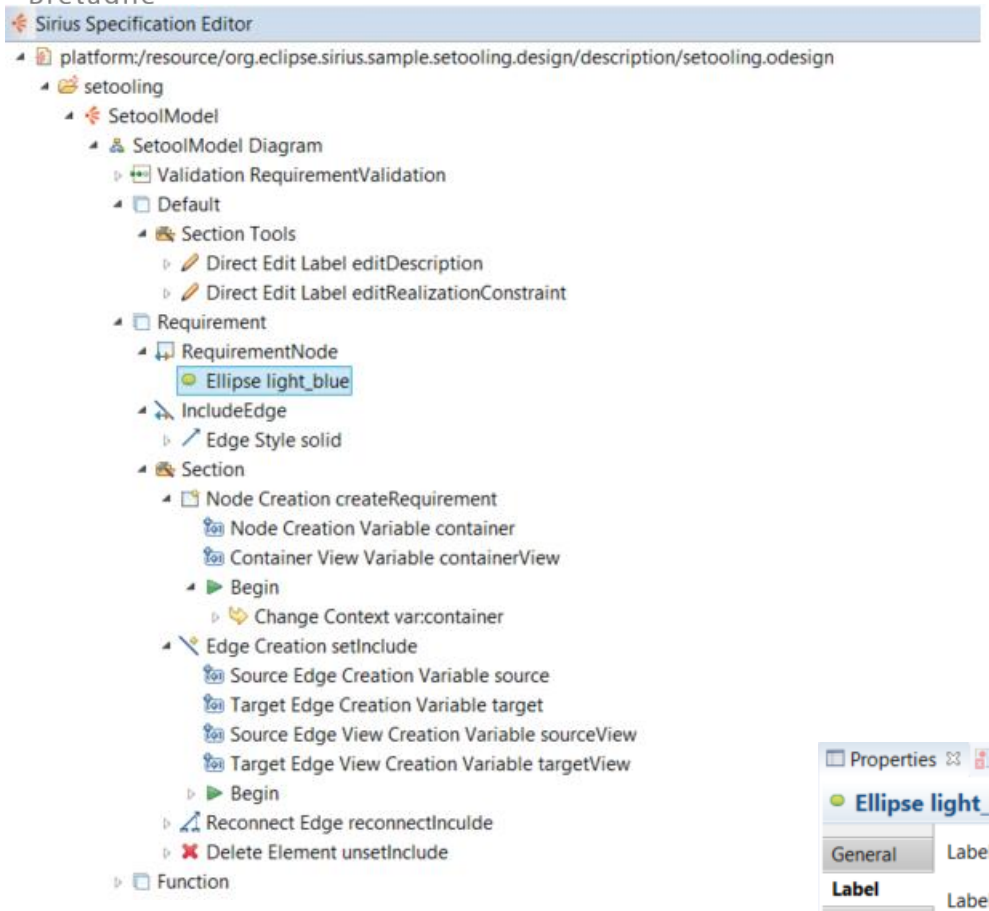
Sirius Eclipse Project
Initiate by Obeo and Thales

Sirius and Graphical syntax

Metamodel of System Engineering language

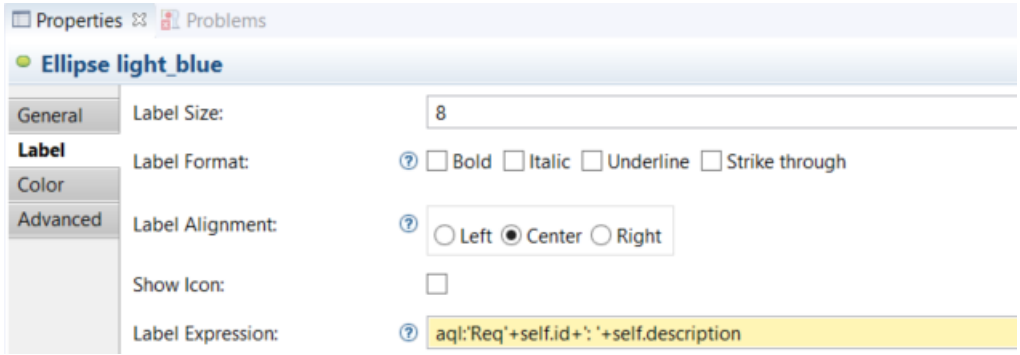


Sirius and Graphical syntax

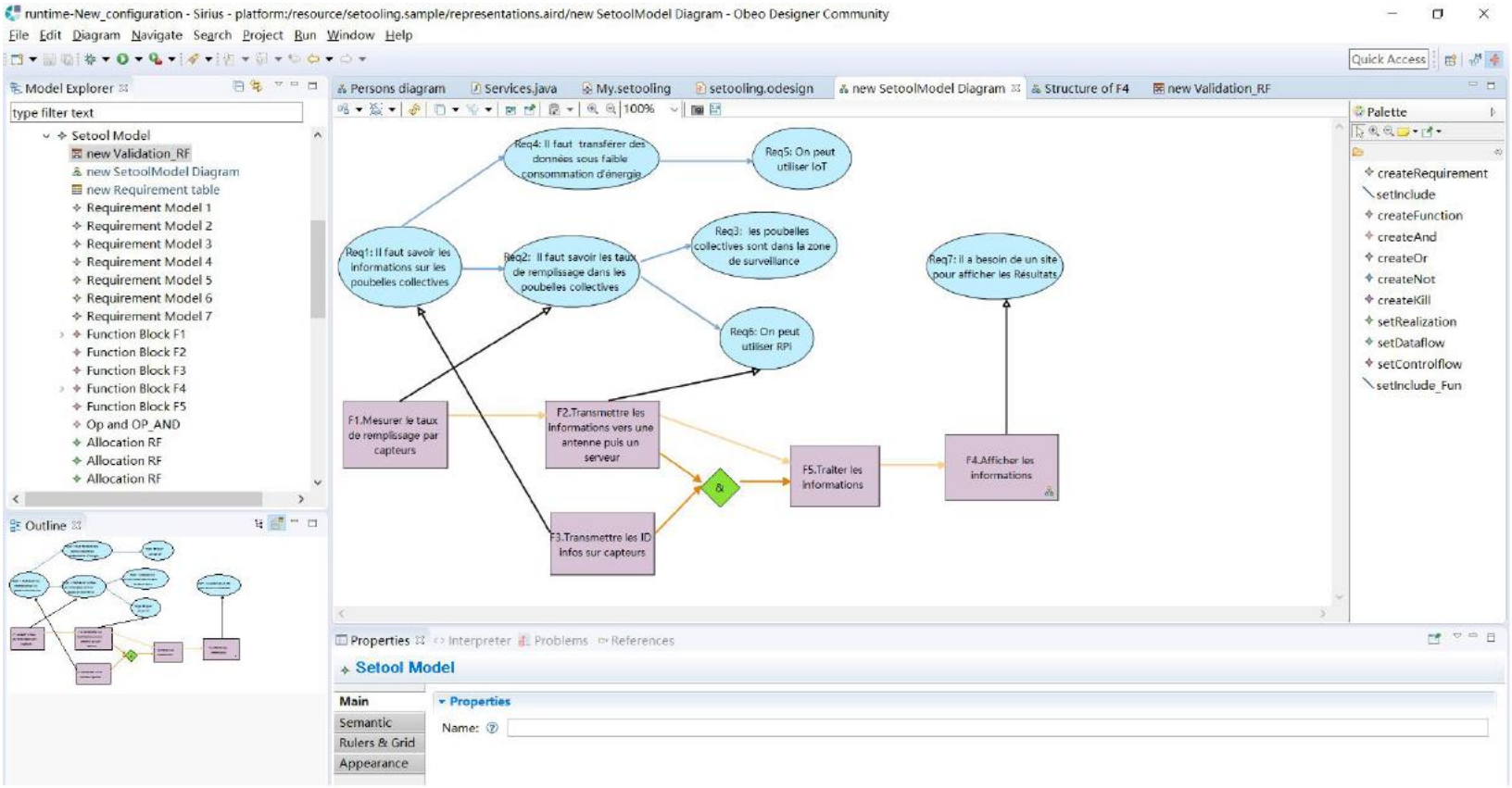


Sirius Odesign model

This model defines the mapping between the metamodel and the graphics elements



Sirius and Graphical syntax



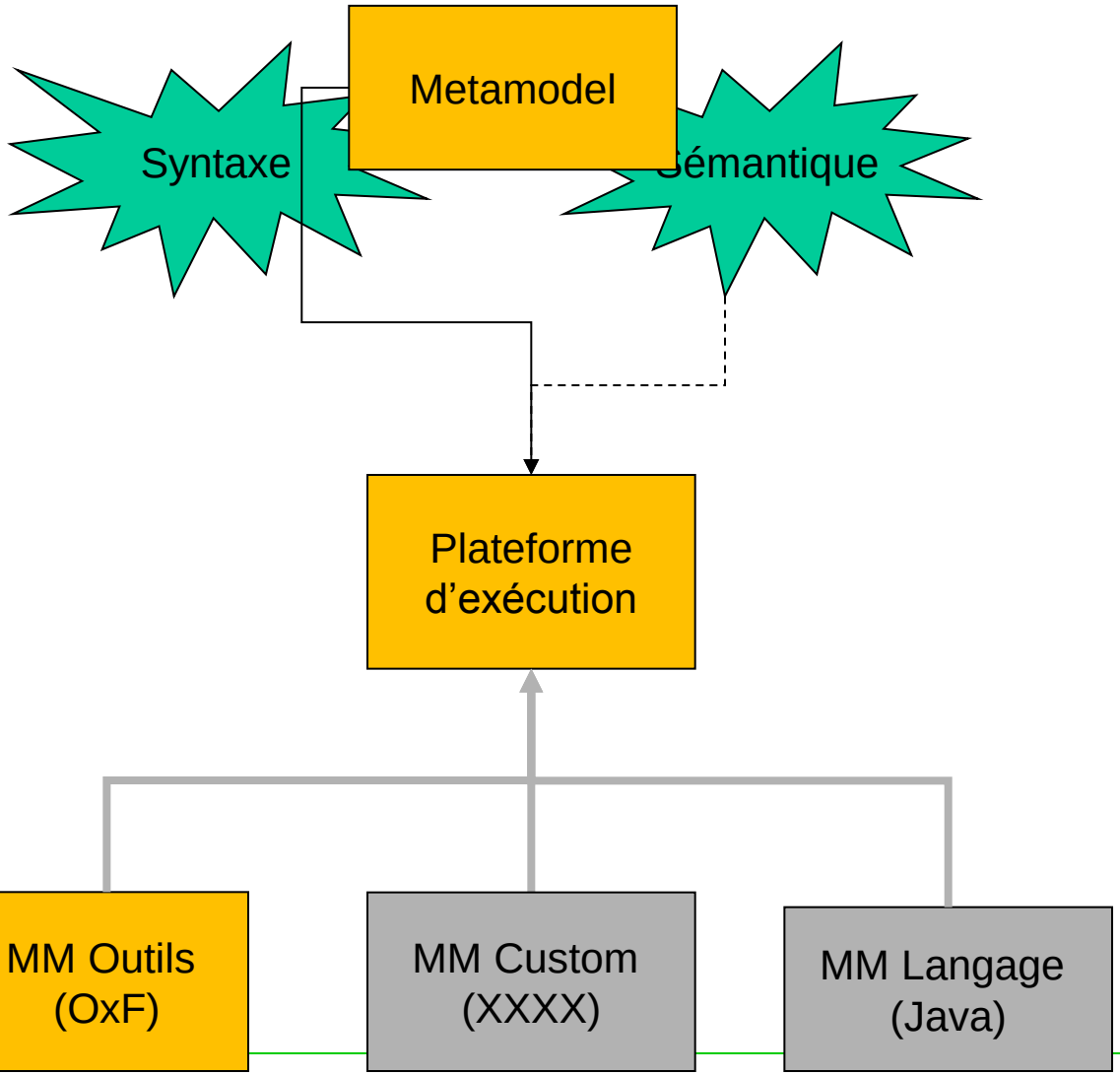
The resulting modeling tool : System Engineering Tool

Sirius and Graphical syntax

Sirius Tooling :

- Mapping between the metamodel and graphical forms
- The Mapping is a Odesign model (based on Ecore model)
- The graphical forms are defined by an Ecore model

- Sirius interpreter interprets the mapping model to create the modeling tool

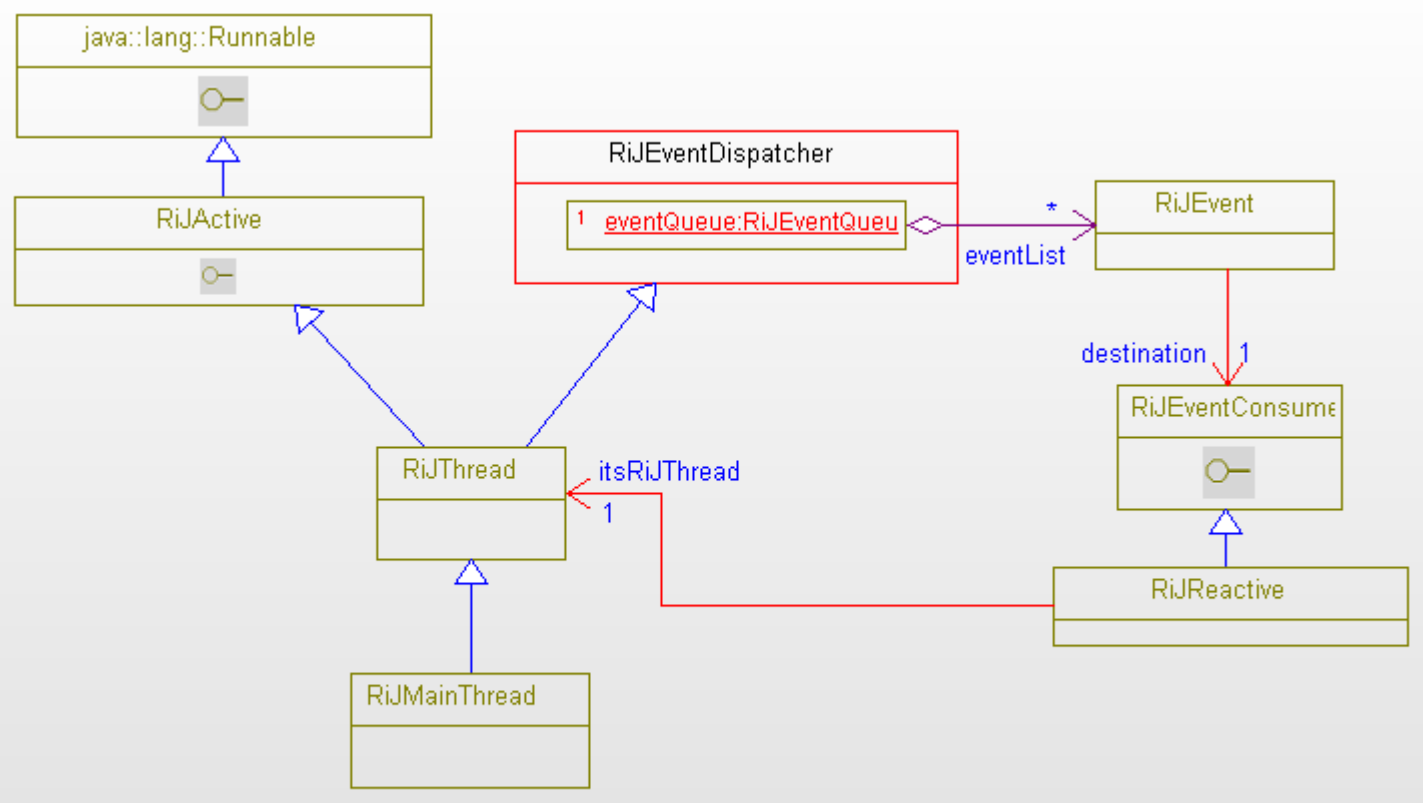

Metamodel
Syntaxe
Sémantique
Plateforme
d'exécution
MM Outils
(OxF)
MM Custom
(XXXX)
MM Langage
(Java)

Exemple Rhapsody

- D Harel, statecharts creator
  - Rhapsody tool
  - Close to UML standard.
  - Concurrency model = Active Objects
  - Communication - Messages Queue FIFO
    - Asynchrous Events - macro obj->GEN(evt) - paramters.
    - Synchronous Messages – high priority on asynchronous msg
    - Method calls
  - Concurrence Model
    - Evt dispatched towards receiver object
    - RTC Consumption.
    - Macro IS_IN(etat) for AND-State.
  - Time with temporal evt
    - Temporal Evt queued in the FIFO.

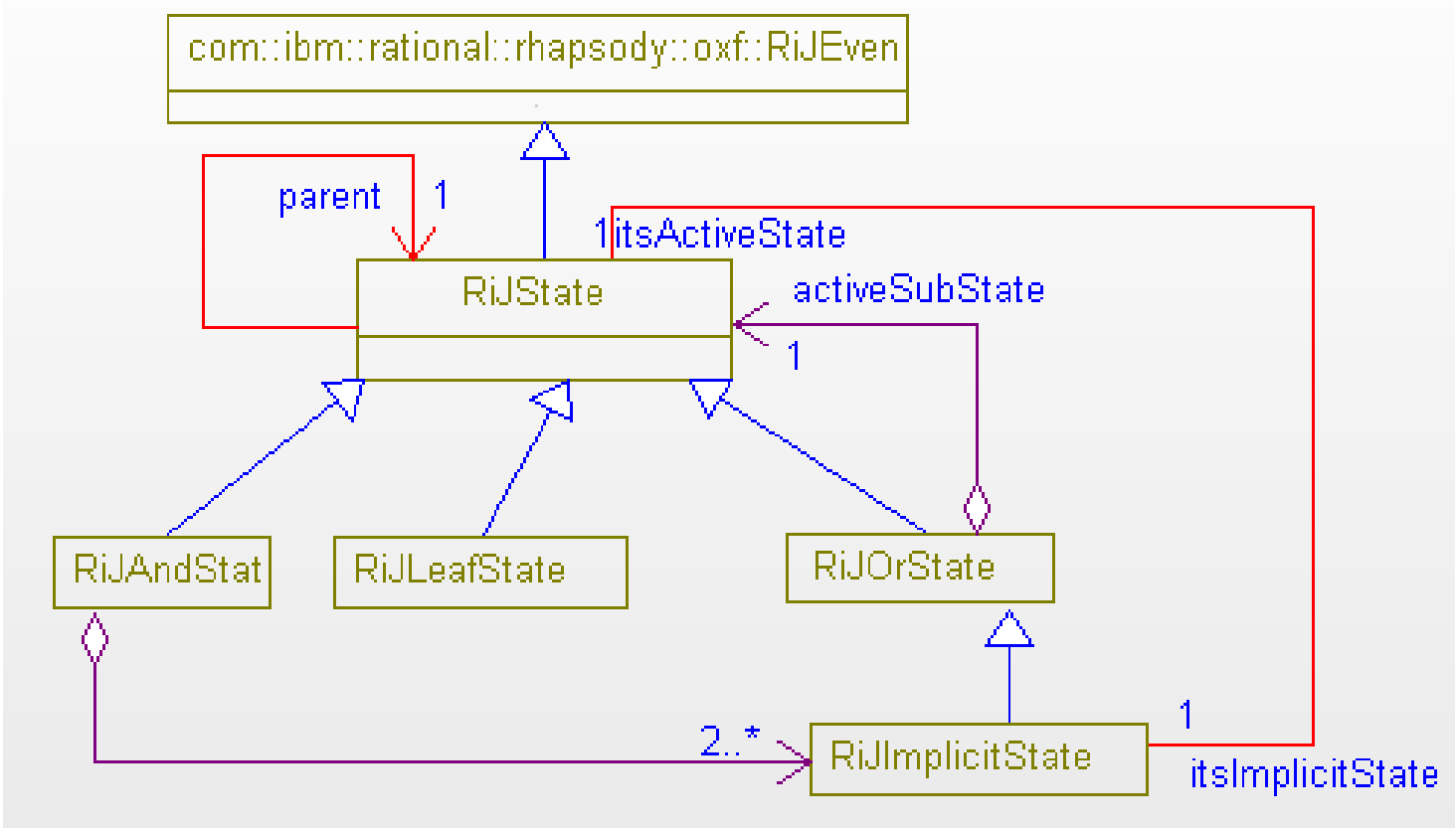
UML Active Objects

- Model of the active objects in Rhapsody



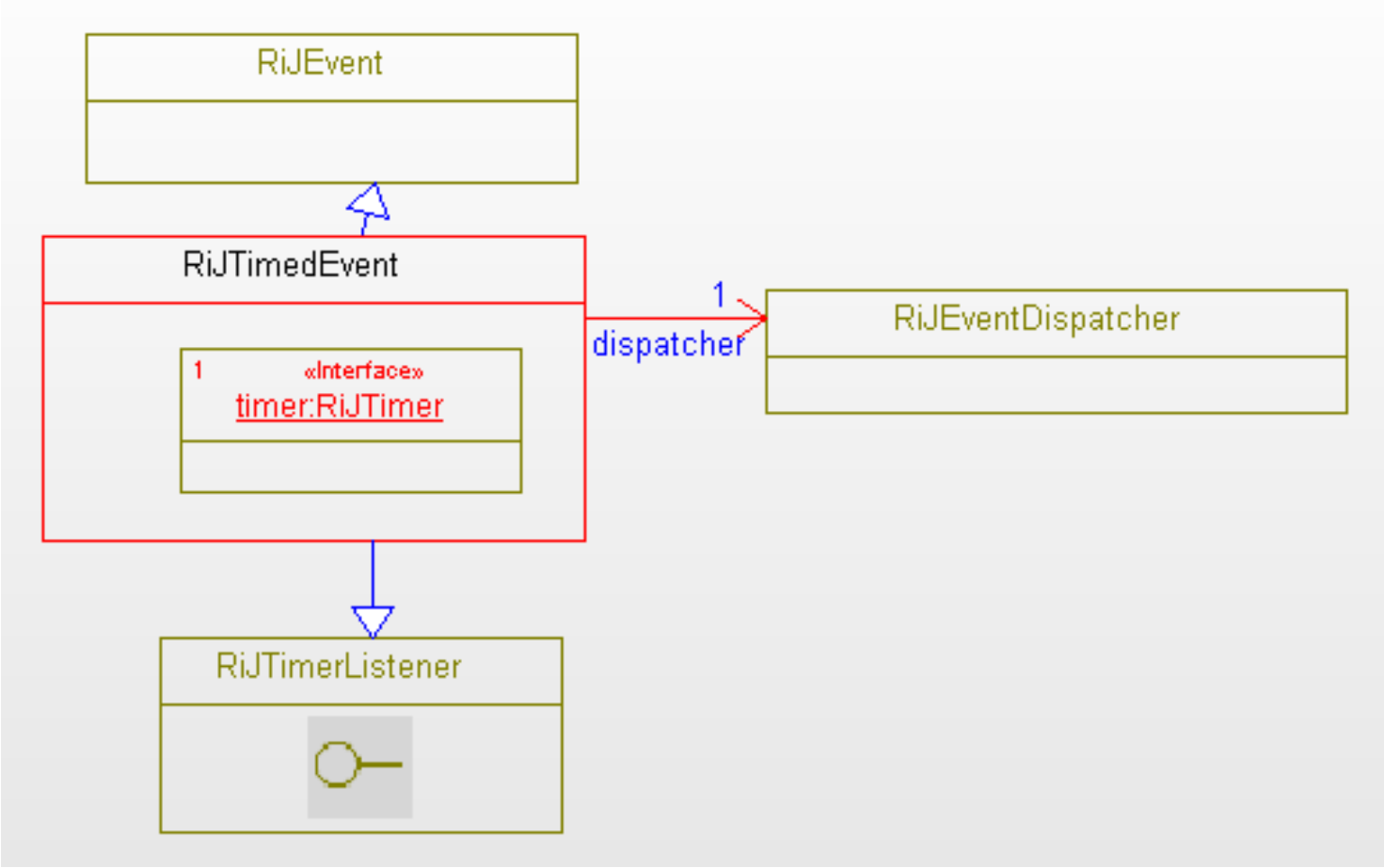
UML Active Objects

- Model of states relative to statecharts implementation



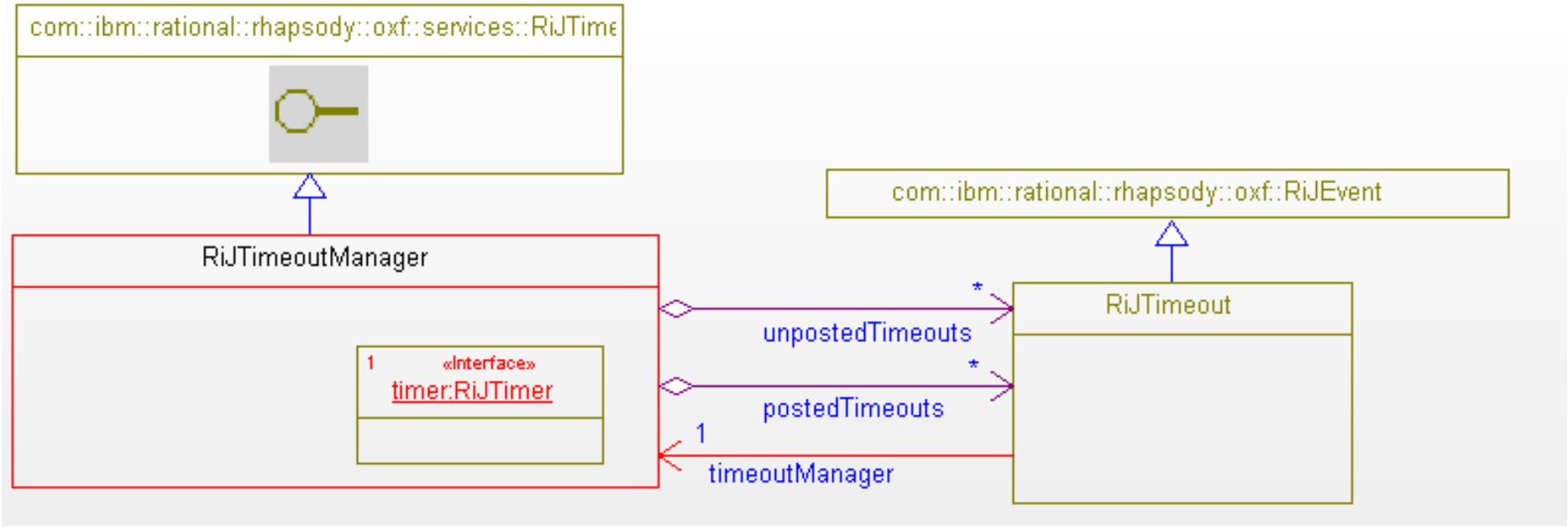
UML Active Objects

- Timed event in the Rhapsody execution model.



UML Active Objects

- Timed event manager in the Rhapsody execution model.



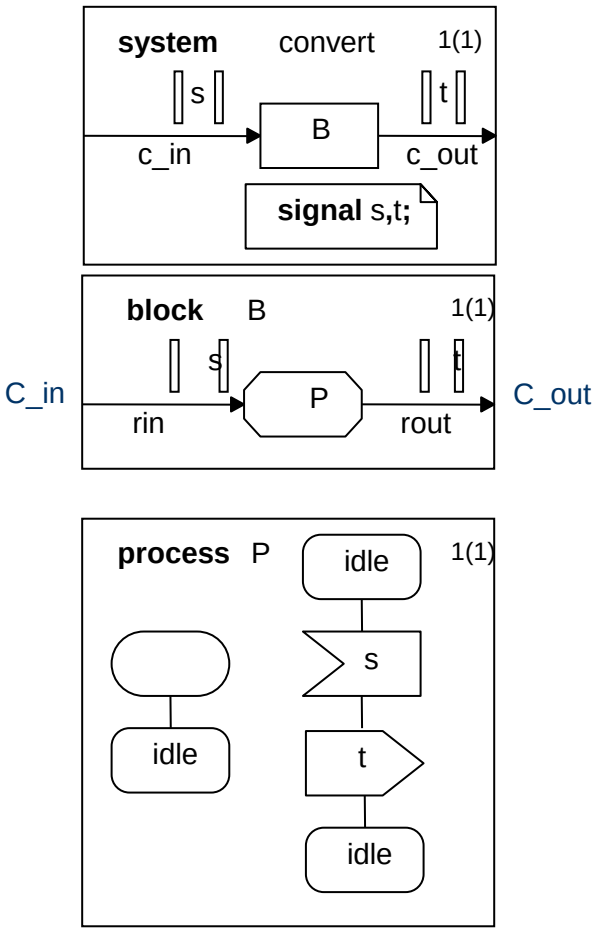
Sujet de Contrôle Continu

- Générateur de code SDL
  - Mise en application d'une démarche de metamodélisation et de production de l'outillage associé
  - Travail à rendre le 11 Décembre par mail
    - Joel.champeau@ensta-bretagne.fr
    - Alain.plantec@univ-brest.fr
  - Générateur du code source SDL à partir d'une instance XML ou Json
    - XML ou Json favorise la réutilisation et l'interopérabilité
    - Applicabilité à tout langage dédié
    - Application du pattern Visiteur

Sujet de Contrôle Continu

- Le langage SDL
  - Langage du domaine des télécommunications pour la spécification des protocoles
    - De la règle à forme (années 80) aux outils informatiques actuels (Pragmadev, etc…)
  - Standard de l'ITU
    - Modèles de protocoles
    - Description structurelle des systèmes
    - Description comportementale des systèmes
      - Machine à états
      - Communication asynchrone
    - Une syntaxe graphique et textuelle

Le langage SDL




```
system convert;
  signal s,t;

  channel c_out nodelay from B to env with t;
  endchannel c_in;

  channel c_in nodelay from env to B with s;
  endchannelc_out;

  block B referenced;

endsystem convert;


block B;
  channel rin nodelay from env to P with s;
  endchannel rin;

  channel rout nodelay from P to env with t;
  endchannel rout;

  process P referenced;

  connect c_out and rout;
  connect c_in and rin;

endblock B;
```

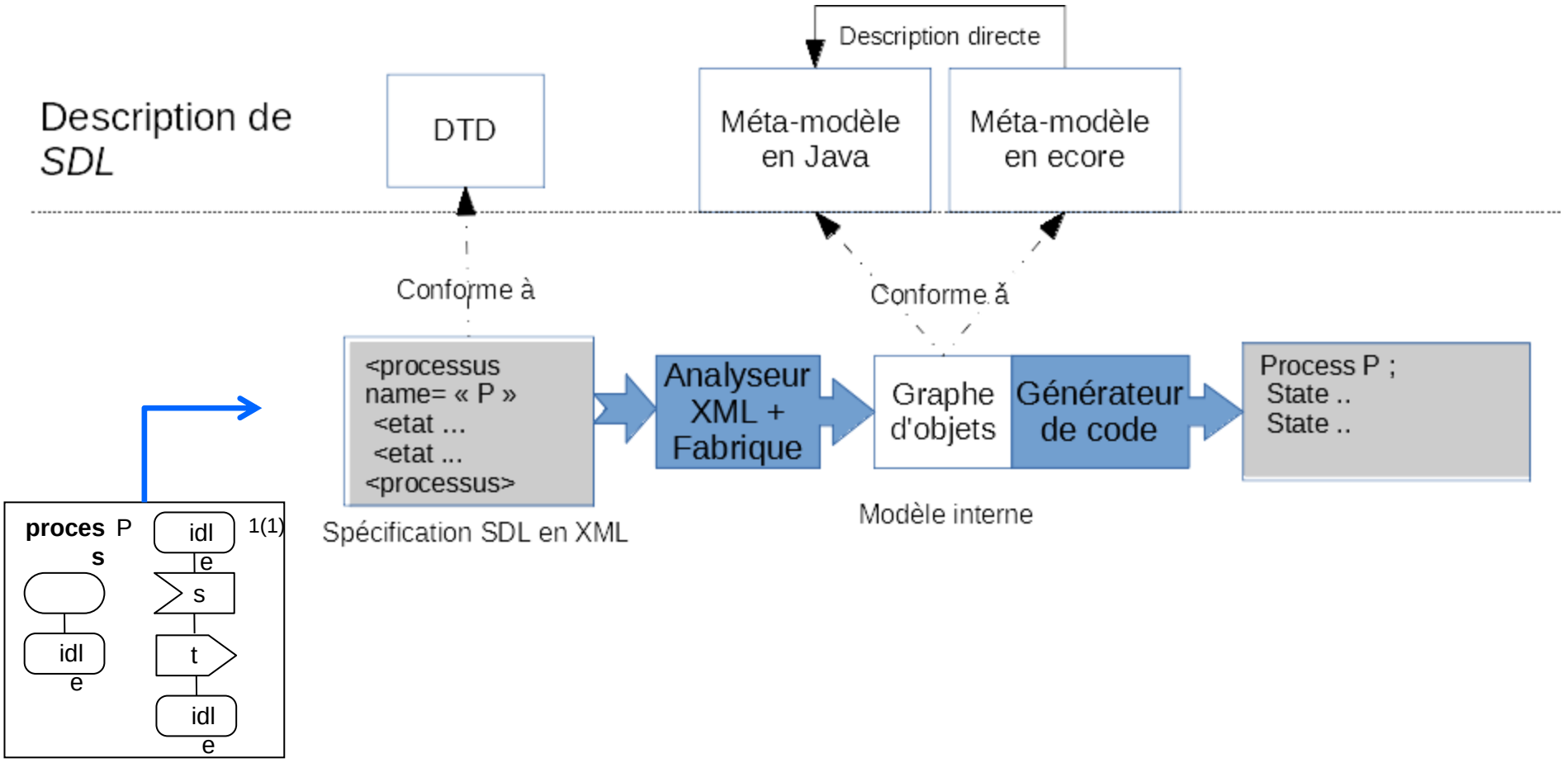
```
process P;
  start;
  nextstate idle;

  state idle;
    input s;
    output t;
      nextstate idle;

  endstate idle;

endprocess P;
```

La chaine de génération de code SDL



La chaine de génération de code SDL

Travail demandé :

- Faire une recherche documentaire du langage SDL
- Dans un premier temps, nous pouvons retenir uniquement le niveau *System* et le niveau *Process* avec son comportement interne pour la communication (état, transition avec réception et expédition de signaux)
  - En fonction de votre avancée, vous pourrez augmenter la description du langage. Soyez agiles, faites plusieurs itérations !!
- Faire en parallèle :
  - Faire le format XML pour une instance du langage
  - Faire le modèle Ecore comme spécification
    - Utilisation Eclipse
    - Création d'un projet Eclipse Modeling Framework – Ecore Model
- Ecrire le code Java du metamodèle
- Faire le parser XML ou Json
  - Voir documentation d'Alain Plantec
- Faire l'instanciation des classes Java du metamodèle à partir du parcours de l'arbre DOM
- Faire le générateur de code SDL (format textuel) à l'aide d'un visiteur
  - voir le cours d'Alain pour le Visiteur