UNIVERSITÉ DE PAU ET DES PAYS DE L'ADOUR

Module Ingénierie des Modèles

Domain Specific Languages
Un exemple de mise en œuvre à travers l'étude de SysML

Nicolas Belloir

Introduction

- Introduction aux DSL
- Construction d'un DSML : un exemple
- Un DSML spécifique : SysML
- Conclusion

Qu'est ce qu'un langage?

- Le **langage** est un ensemble de signes (vocaux, gestuels, graphiques, tactiles, olfactifs, etc.) doté d'une **sémantique**, et le plus souvent d'une **syntaxe** [...]. Plus couramment, le langage est un **moyen de communication** [WIKI].

Les DSL's

- Domain Specific Languages (fabriques à langage dédiés)
  - Générique (UML) Versus Spécifique
    - Meilleure solution à un plus petit ensemble de problèmes
  - Une définition
    - « A DSL is a programming language or executable specification language that offers, through appropriate notations and abstractions, expressive power focused on, and usually restricted to, a particular problem domain ».
    - A rapprocher des Domain Specific Models
  - Exemple :
    - Informatique : YACC, PIC, SQL, HTML …
    - Autres domaines : médecine, animation 3D, communication protocoles, business protocoles …

Avantages

- Haut niveau d'abstraction : compréhensible par des experts métiers
- Langage concis et bien documenté, réutilisable
- Amélioration de la productivité, fiabilité, maintenabilité, portabilité
- Incluent la connaissance du domaine (conservation)
- Permettent la validation et l'optimisation au niveau domaine

Risques

- Coût important :
  - de développement et d'implémentation élevés
  - Pour la formation des utilisateurs
- Faible disponibilité des DSL
- Difficulté à limiter un bon champ d'action
- Equilibre entre DSL et langages génériques délicat à trouver
- Perte potentiel d'efficacité p/r à un code réalisé « à la main »

Méthodologie de conception

- Analyse
  - a) Identifier le domaine
  - b) Récupérer la connaissance associée
  - c) Rassembler cette connaissance en quelques groupes sémantiques d'opérations et de notions
  - d) Concevoir le DSL pour qu'il décrive de manière concise les applications du domaine
- Implémentation
  - e) Construire une librairie qui implémente les notions sémantiques
  - f) Concevoir et implémenter un compilateur transformant les programmes DSL en une séquence d'appels à la librairie
- Use
  - G) Ecrire des programmes DSL

Introduction

- Introduction aux DSL
- Construction d'un DSML : un exemple
- De UML à SysML
- Conclusion

Module Ingénierie des Modèles

Construction d'un DSML : un exemple

Réalisé par:

Liuppa Labs, Movies Team, **University of Pau**, France

***Neomades** Cie*, Bidart, France

Estia, Bidart, France

Problem & Motivation

- Mobile Application Development Ecosystem's
  - Smartphones:
    - iPhone
    - Android
    - Bada
    - Windows Phone
    - …
  - Mass market:
    - Java ME

Problem & Motivation

- Smartphones:
  - Development environment
  - Programmaing languages, APIs
  - …
- Java ME:
  - Standards: MIDP (1.0, 2.0, 2.1), CLDC (1.0, 1.1)…
  - Optional APIs: File IO, PIM, Bluetooth…
  - Implementation diversity: different interpretation, bugs, security policy
  - Specific hardware (screen size, inputs methods…)

→ How many specific versions of each application? (Porting Tools!)

→ **How to ensure end-user quality assurance?**

Problem & Motivation

- Application tests are made on real handsets ~ 250 in average
- Visual and sonorous interpretation of the application's behavior
- Costly:
  - Time
  - Handsets
  - Testers' Team
  - From scratch for each application

Problem & Motivation - Test steps sample

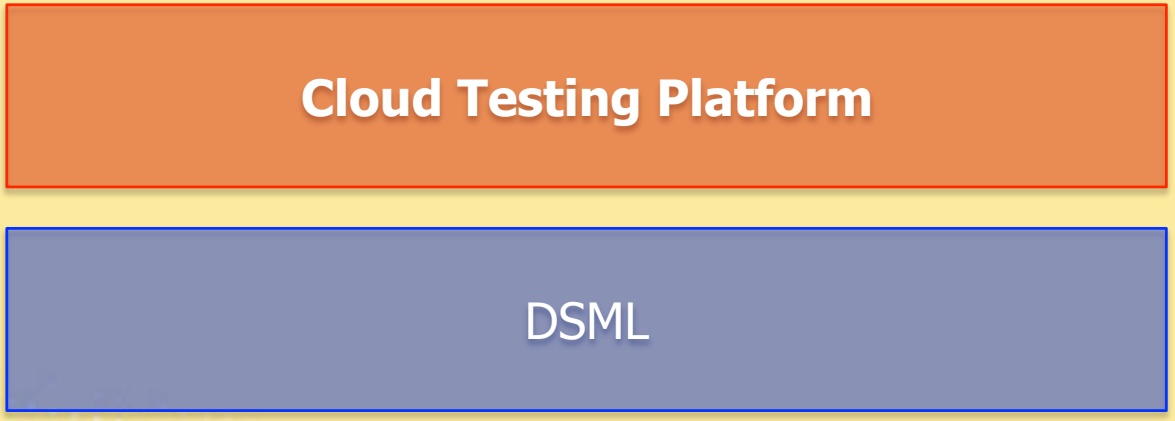
35	6	Perform an incoming call in-game in **pause menu**		-	The application is suspended and goes into **pause** state	
36		a	**Reject** call	Red handset key	You come back to the **pause menu**	
37		b	**Accept** call	Green handset key	As soon as you end your call, you come back to the **pause menu**	
38		c	**End call** with second device	Red handset key	You come back to the **pause menu**	
39						
40	B - SMS / MMS					Results
41	Steps				Keys	Expected Result
42	1	Launch the game			-	-
43	2	Send a **SMS / MMS** with other device while game **is loading**			-	You receive a **notification** (sound or icon) that you have received a SMS / MMS and the game **continues normally**
44	3	Send a **SMS / MMS** with other device while on **Main Menu**			-	You receive a **notification** (sound or icon) that you have received a SMS / MMS and the game **continues normally**
45	4	Send a **SMS / MMS** with other device **in-game**			-	You receive a **notification** (sound or icon) that you have received a SMS / MMS and the game **continues normally**
46	4	Send a **SMS / MMS** with other device in-game **in pause menu**			-	You receive a **notification** (sound or icon) that you have received a SMS / MMS and the game **stays on the pause menu**
47						
48	C - Red button					Results
49	Steps				Keys	Expected Result
50	1	Launch the game			-	-
51	2	Press **red button** while game is **loading**			Red handset key	Game is **suspended**, and when you resume the application, game **continues loading normally**

Alternatives?

Module Ingénierie des Modèles

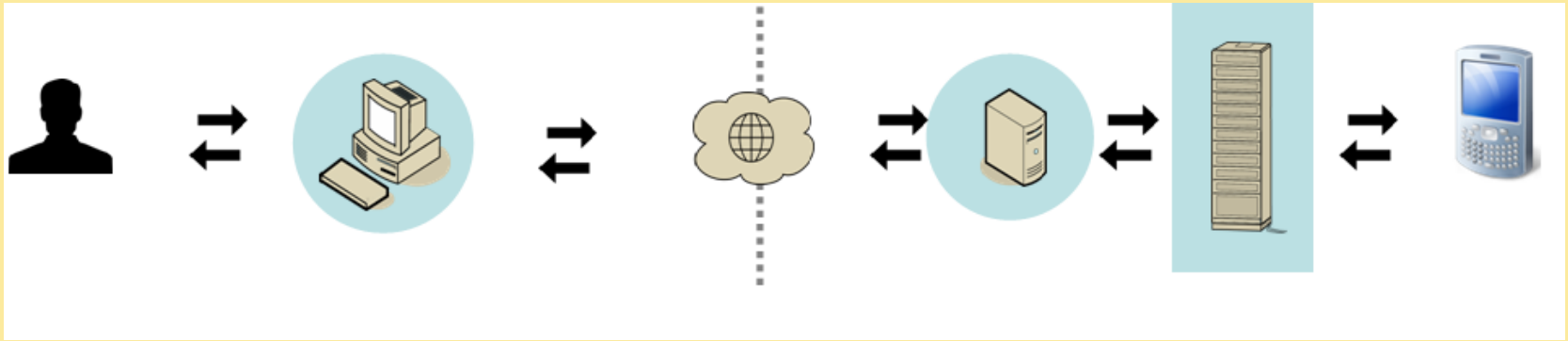
Alternative

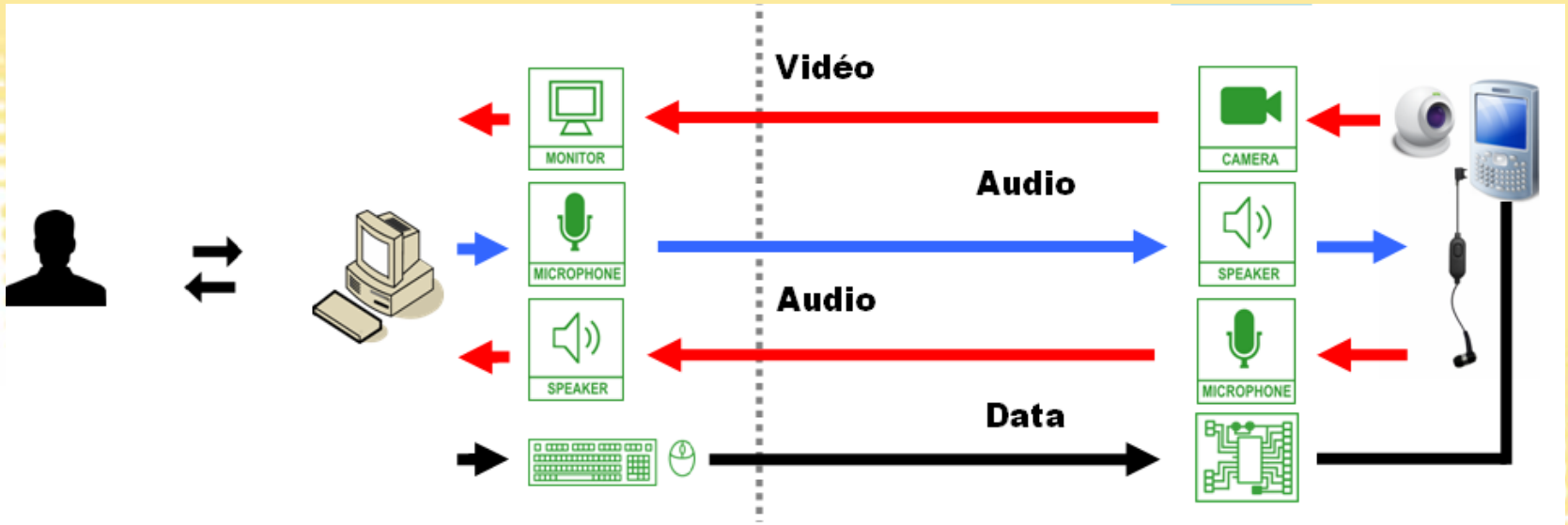
- Provide more **efficient environment**



- Manage **variability**

Cloud Testing - Architecture





Cloud Testing – the Neomades prototype

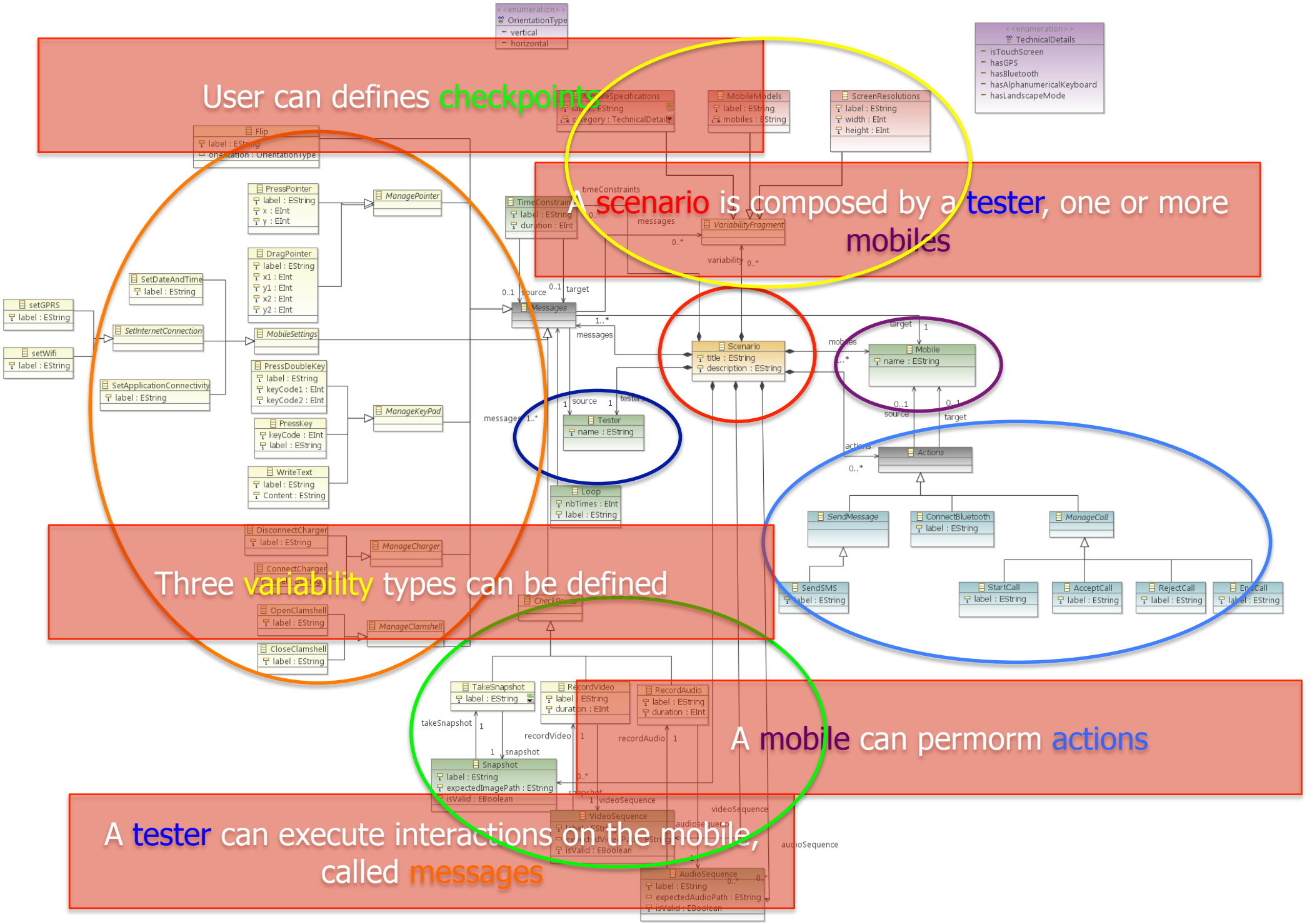
- Internal prototype for our research

The DSML - Description

- A Modeling Language to **design test scenarios**
- Resembling UML Sequence diagrams
  - Lifelines: Tester or Mobiles
  - Messages:
    - Tester → Mobile: press a key, press the pointer…
    - Mobile → Mobile: call, send SMS…
- No programming skills are required
- A scenario can be adapted for another application

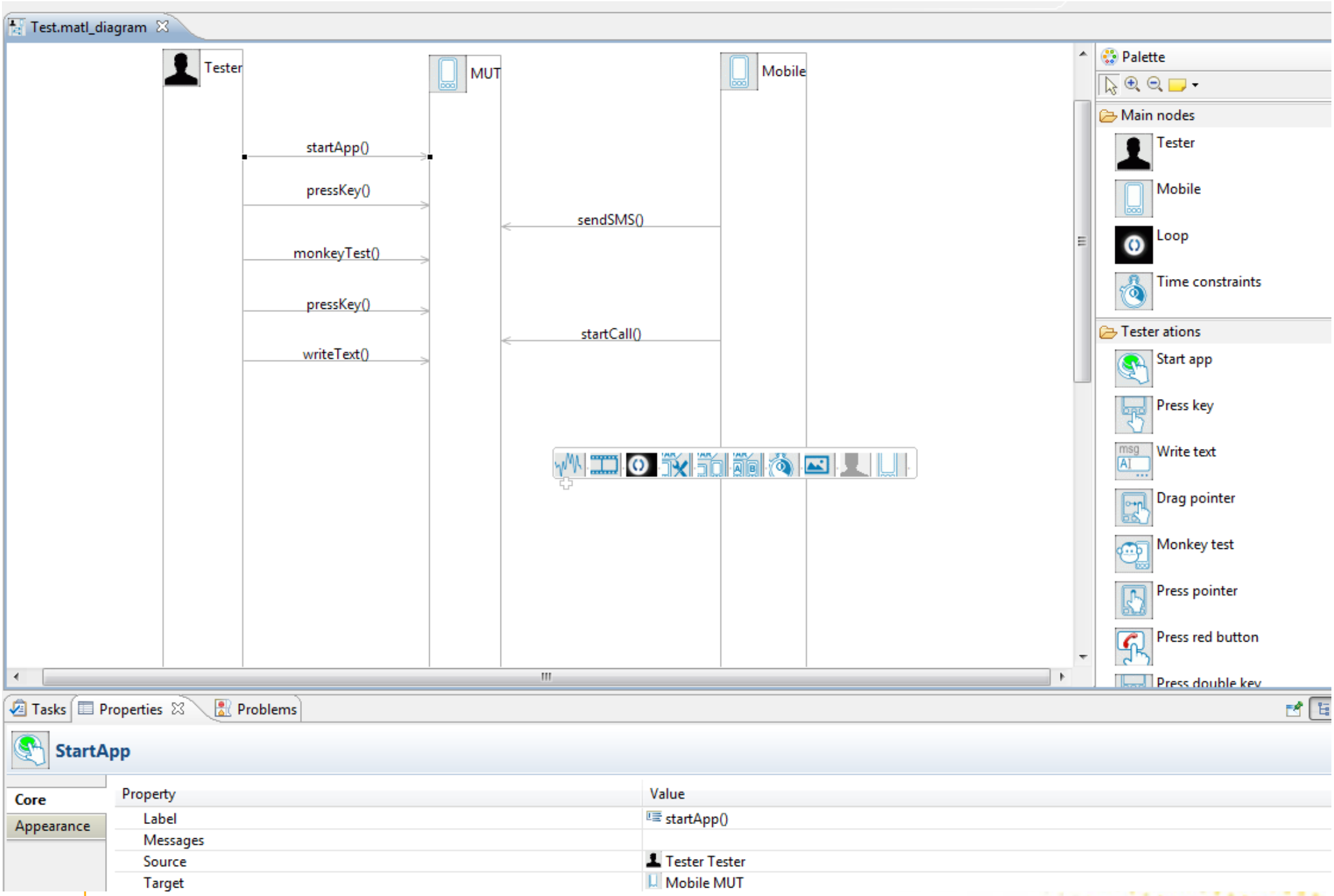
The DSML - Description

- Basic domain elements
  - Tester
  - Mobile under test
  - A second Mobile (optional)
  - Simple actions: press a key…
- **Variabiliy Management**
- Interruptions
- Automates actions
- CheckPoints and results validation


User can defines checkpoints
A scenario is composed by a tester, one or more mobiles
Three variability types can be defined
A mobile can permorm actions
A tester can execute interactions on the mobile, called messages
<<enumeration>>
OrientationType
vertical
horizontal
<<enumeration>>
TechnicalDetails
isTouchScreen
hasGPS
hasBluetooth
hasAlphanumericalKeyboard
hasLandscapeMode
MobileSpecifications
label : EString
category : TechnicalDetails
MobileModels
label : EString
mobiles : EString
ScreenResolutions
label : EString
width : EInt
height : EInt
Flip
label : EString
orientation : OrientationType
PressPointer
label : EString
x : EInt
y : EInt
ManagePointer
DragPointer
label : EString
x1 : EInt
y1 : EInt
x2 : EInt
y2 : EInt
SetDateAndTime
label : EString
setGPRS
label : EString
setWifi
label : EString
SetInternetConnection
MobileSettings
SetApplicationConnectivity
label : EString
PressDoubleKey
label : EString
keyCode1 : EInt
keyCode2 : EInt
Presskey
keyCode : EInt
label : EString
ManageKeyPad
WriteText
label : EString
Content : EString
DisconnectCharger
label : EString
ManageCharger
ConnectCharger
label : EString
OpenClamshell
label : EString
CloseClamshell
label : EString
ManageClamshell
timeConstraints
TimeConstraint
label : EString
duration : EInt
messages
VariabilityFragment
variability
source
target
Messages
Scenario
title : EString
description : EString
mobiles
Mobile
name : EString
tester
Tester
name : EString
Loop
nbTimes : EInt
label : EString
Actions
SendMessage
ConnectBluetooth
label : EString
ManageCall
SendSMS
label : EString
StartCall
label : EString
AcceptCall
label : EString
RejectCall
label : EString
EndCall
label : EString
CheckPoints
TakeSnapshot
label : EString
RecordVideo
label : EString
duration : EInt
RecordAudio
label : EString
duration : EInt
takeSnapshot
recordVideo
recordAudio
snapshot
Snapshot
label : EString
expectedImagePath : EString
isValid : EBoolean
videoSequence
VideoSequence
isValid : EBoolean
audioSequence
AudioSequence
label : EString
expectedAudioPath : EString
isValid : EBoolean

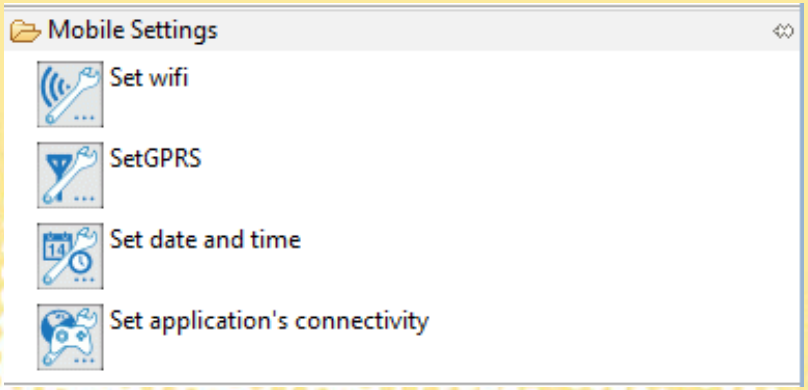
The DSML - Basic domain elements

- Tester
- Mobile under test
- A second Mobile (optional)
  - Simple actions:
    - press a key
    - press pointer
    - rotate the phone
    - …


Test.matl_diagram
Tester
MUT
Mobile
startApp()
pressKey()
sendSMS()
monkeyTest()
pressKey()
startCall()
writeText()
Palette
Main nodes
Tester
Mobile
Loop
Time constraints
Tester ations
Start app
Press key
Write text
Drag pointer
Monkey test
Press pointer
Press red button
Press double key
Tasks
Properties
Problems
StartApp
Core
Appearance
Property
Value
Label
startApp()
Messages
Source
Tester Tester
Target
Mobile MUT

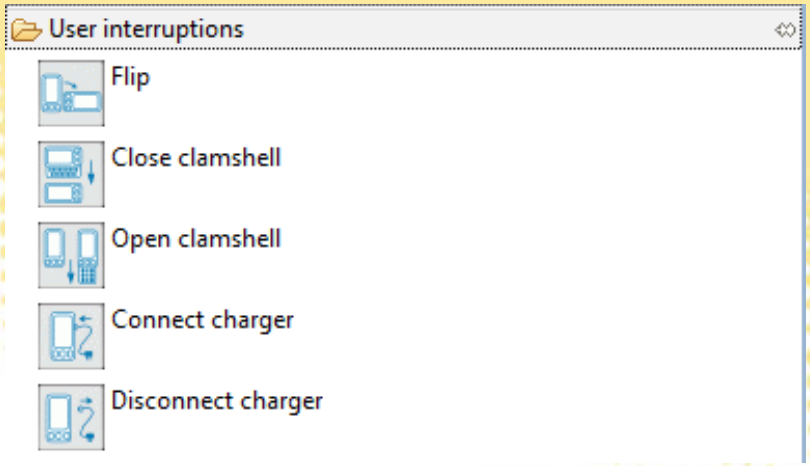
The DSML - Automated actions

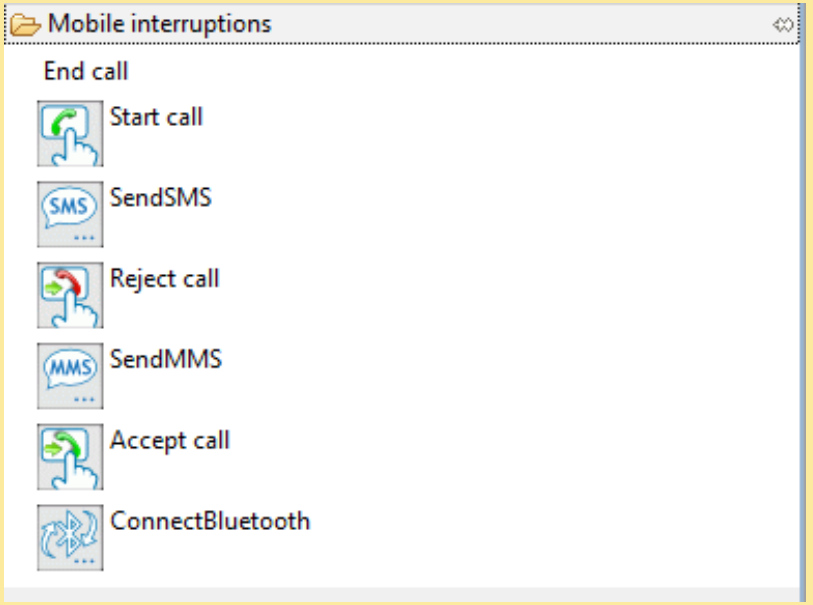
- Write Text
- Send SMS
- Set Wifi connection
- Set date and Time
- …



The DSML - Interruptions

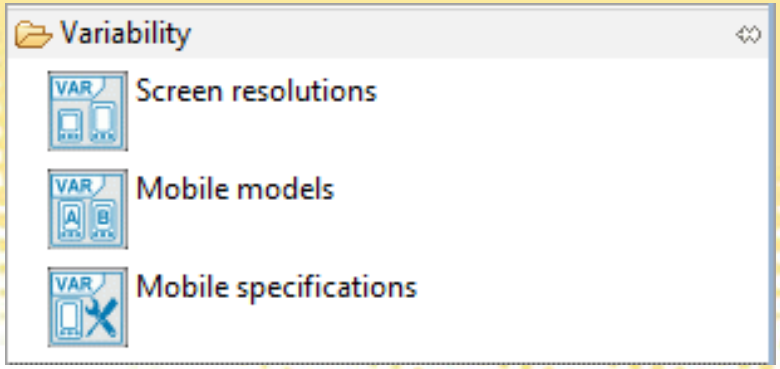
- Test application behavior
  - Send SMS
  - Bluetooth connection
  - Incoming call
  - …

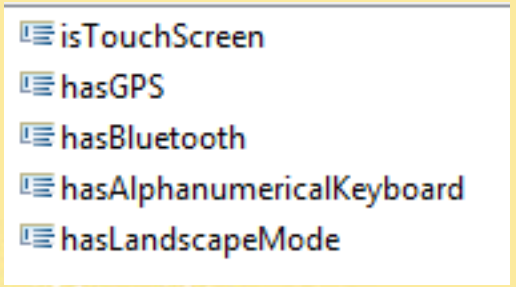




The DSML - Variability Management

- The same application can vary depending on:
  - Different mobile screen resolutions
  - Mobile models: Nokia N95, Samsung Galaxy S…
  - Mobile specifications: has GPS, has Bluetooth…





The DSML - Variability Management

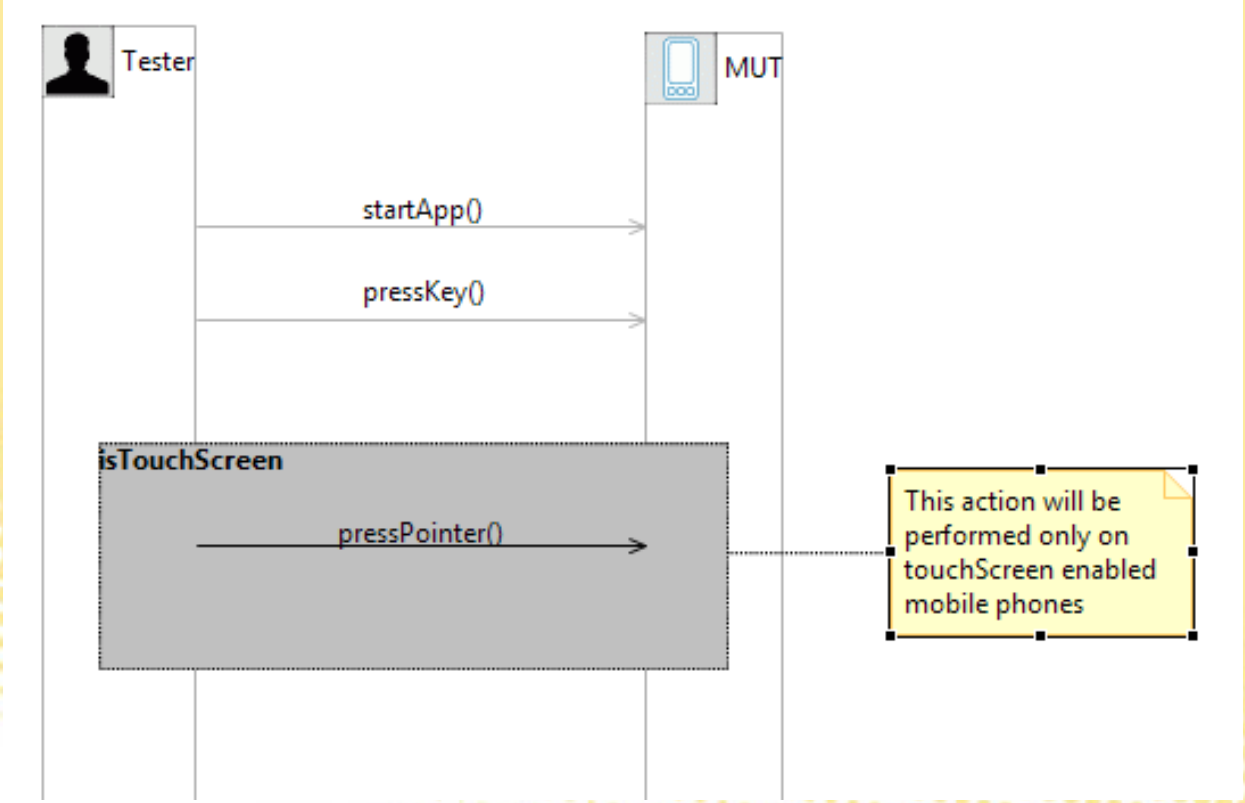
- In practice:
  - Create a variability point
  - Assign one ore more action to this variability point
- A scenario can contains many variability points
- The execution paths are managed on the testing bed thanks to a database (which phone has GPS…)

→ Actions will be executed only on the targeted phones.

→ **Only one scenario for all the devices**

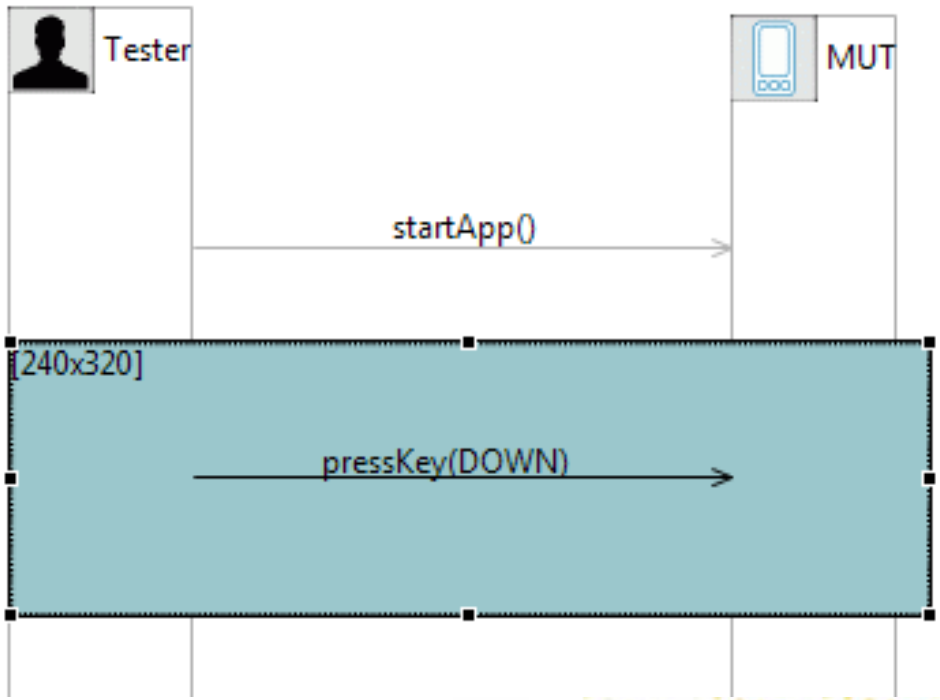
The DSML - Example : Mobile specifications

- Actions assigned to touch screen enabled phones



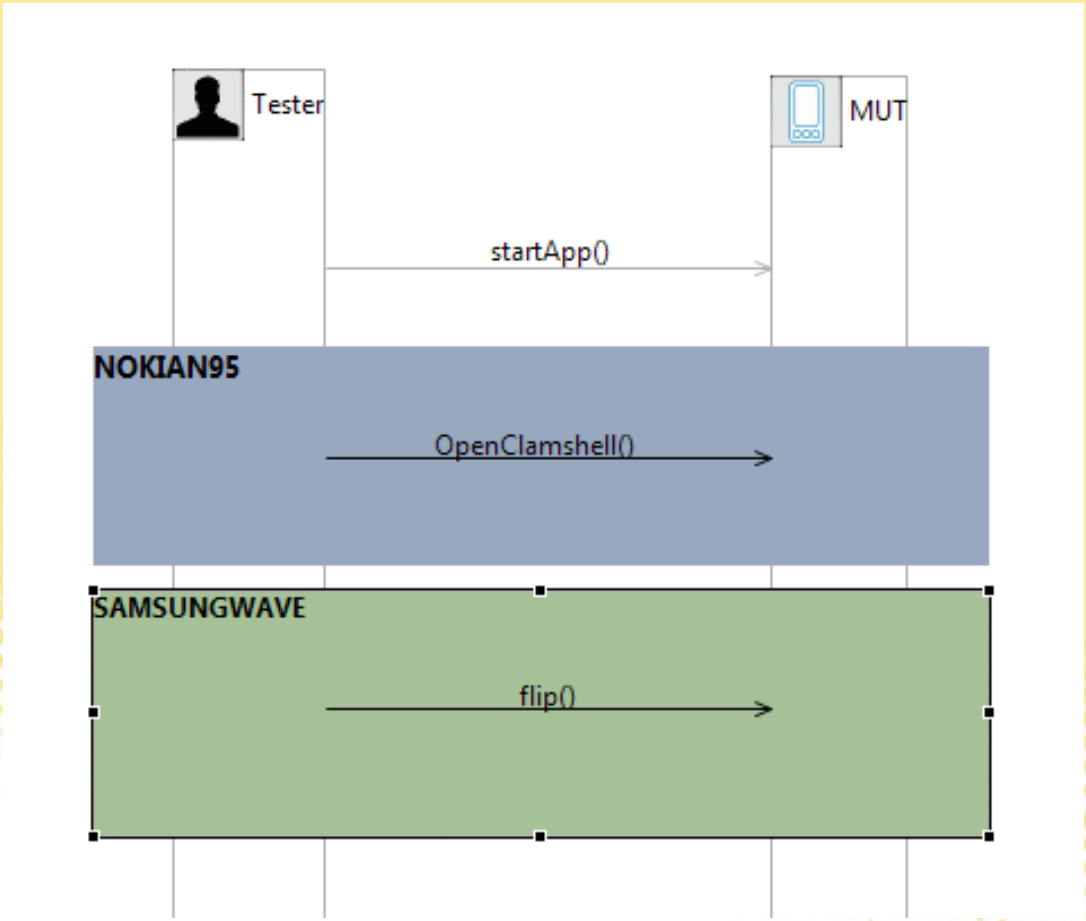
The DSML - Example : Screen Size

- Actions performed only on mobile phones with a 240x320 resolution



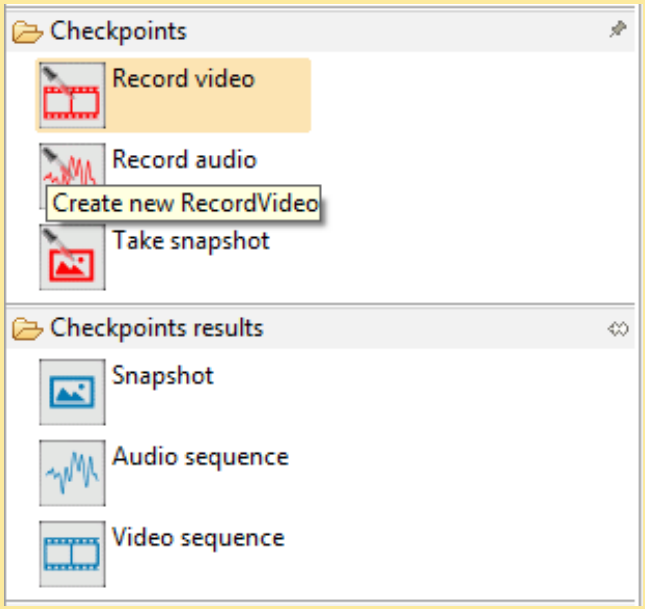
The DSML - Example : Mobile Models

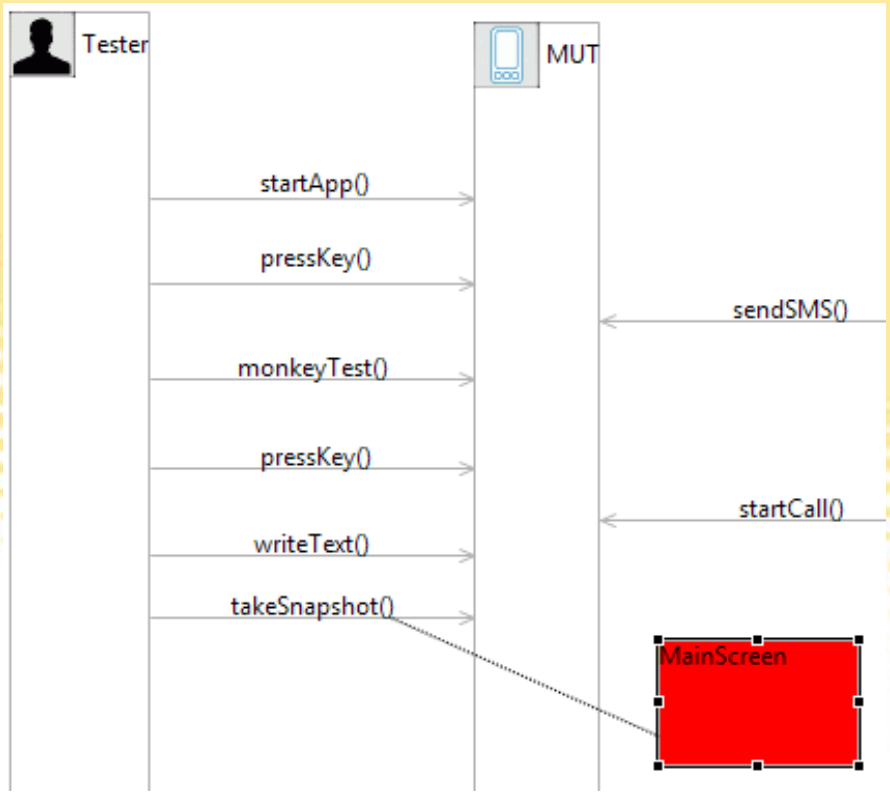
- Assign actions only to one or more specific device



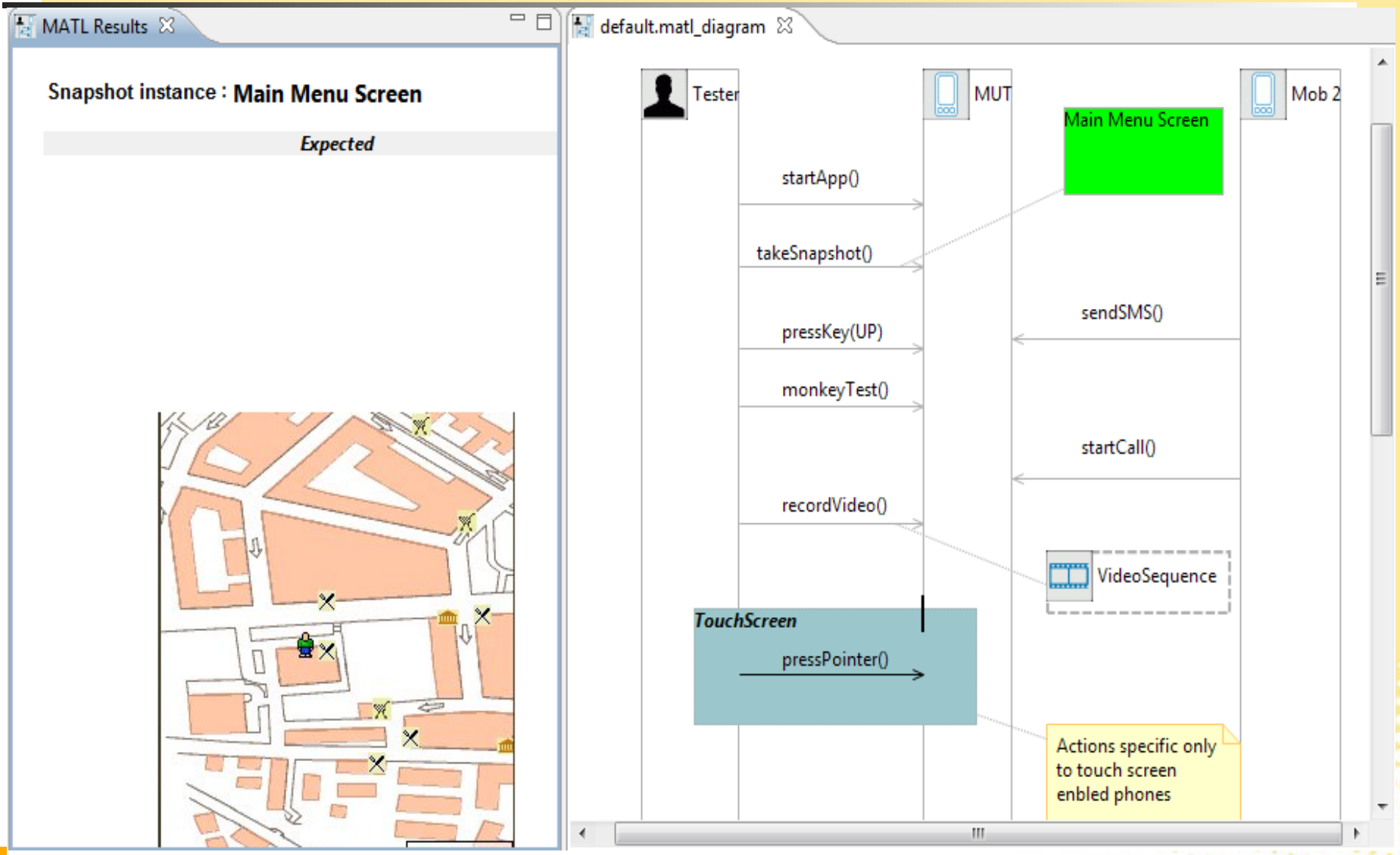
The DSML - Check Points

- Take a Snapshot
- Record an Audio Sequence
- Record a video sequence





The DSML - Result validation



The DSML - Tools

- Eclipse Modeling Framework
  - Metamodel of the DSL
- Graphical Modeling Framework
  - Modeler
  - + Eclipse environment (drag & drap, copy/paste, appearance…)
  - - Bugs!
- Eclipse RCP and plugins
  - Link to the testing bed for scenario execution

Introduction

- Introduction aux DSL
- Construction d'un DSML : un exemple
- Un DSML spécifique : SysML
  - Présentation générale de SysML
  - Modélisation structurelle
  - Modélisation dynamique
  - Modélisation transverse
  - Réflexion sur SysML
- Conclusion

Module Ingénierie des Modèles

Un DSL spécifique : SysML

L’ Ingénierie Système

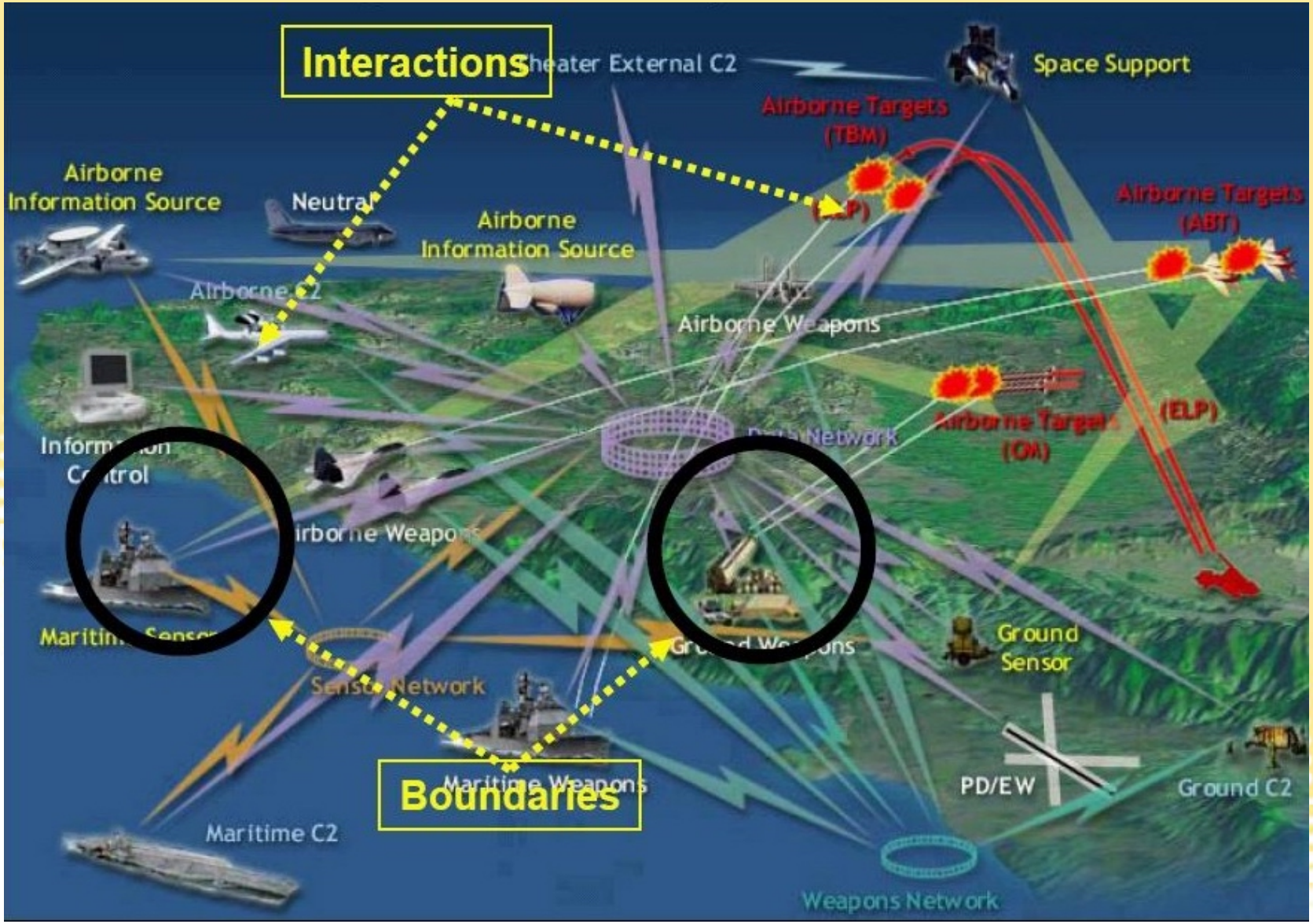
- L'ingénierie système (IS)
  - une démarche méthodologique
  - interdisciplinaire
  - pour maîtriser la conception des **systèmes complexes**
  - [AFIS, WIKI].

A Complex System

- Set of human and material elements composed of various technologies
  - Computer, Hydraulic, Electronic,…
- Integrated to provide services to it environment corresponding to the system finality
- Interacting between themselves and the environment

A complex system is very different of a simple software system

Système complexe et systèmes de systèmes



Module Ingénierie des Modèles

Dans un système …

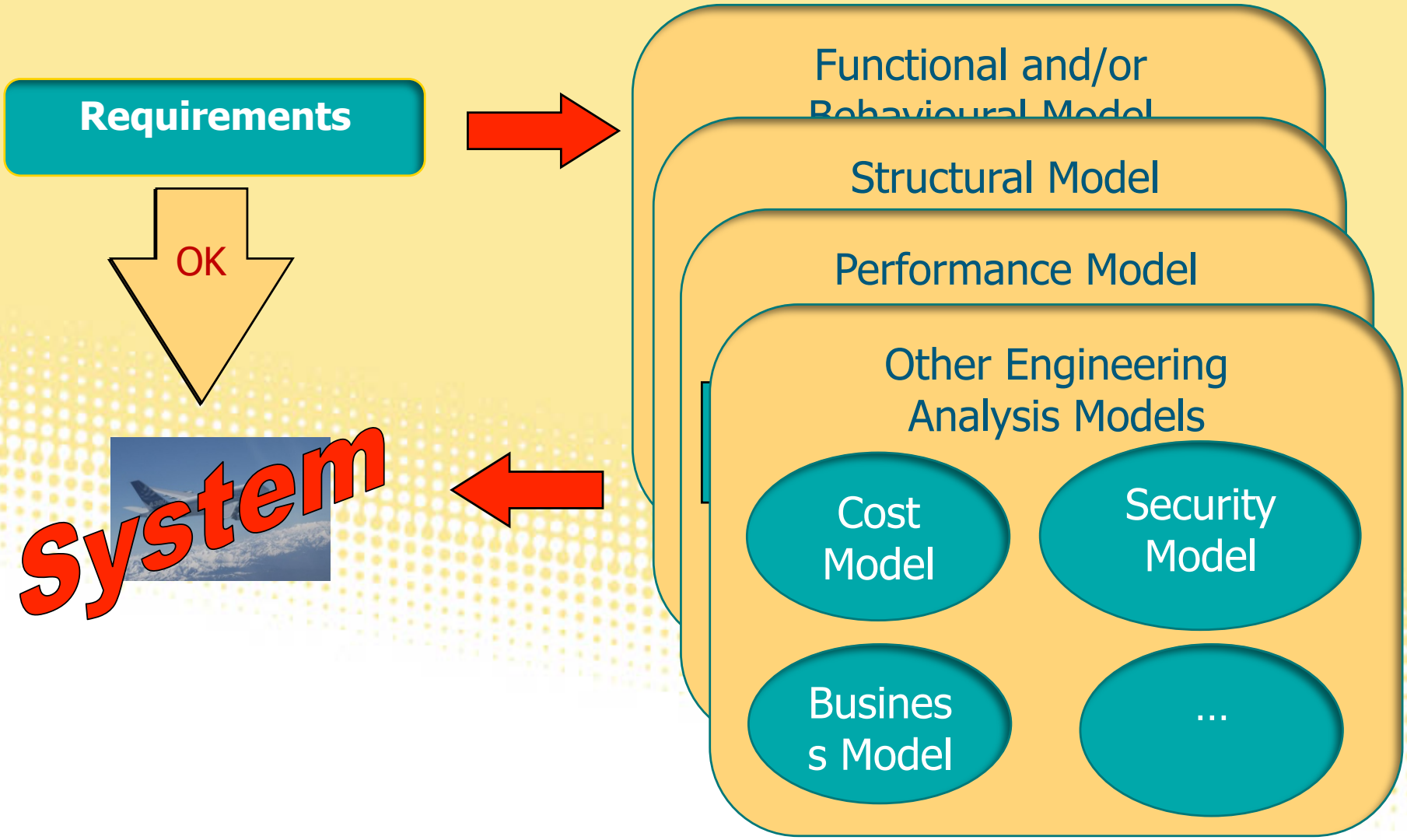
- Propriétés nouvelles résultant des **interactions entre ses constituants**.

- Un système doit :
  - Atteindre les **comportements recherchés**
  - Maintenir les **comportements émergents** non intentionnels dans des limites acceptables.

Démarche méthodologique : un processus itératif

- Exploration du problème
  - Définition de **sous-systèmes** et constituants
  - Sous forme d'**ensembles d'exigences**
- Conception conduisant à des modèles constructifs
  - Architecture fonctionnelle et architecture de constituants
  - Exigences spécifiées de réalisation, d'intégration, de vérification et validation ainsi que de maintenance.

System Modeling

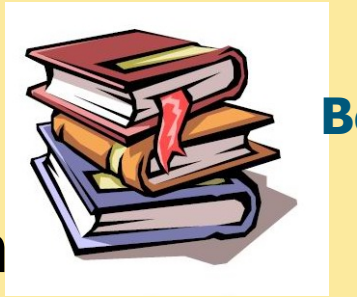


SE Practices for Describing Systems

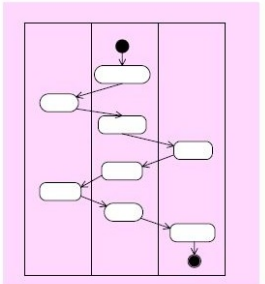
Generate lot of writing work

Not adapted to discuss within a multi-domain team

nition



Before



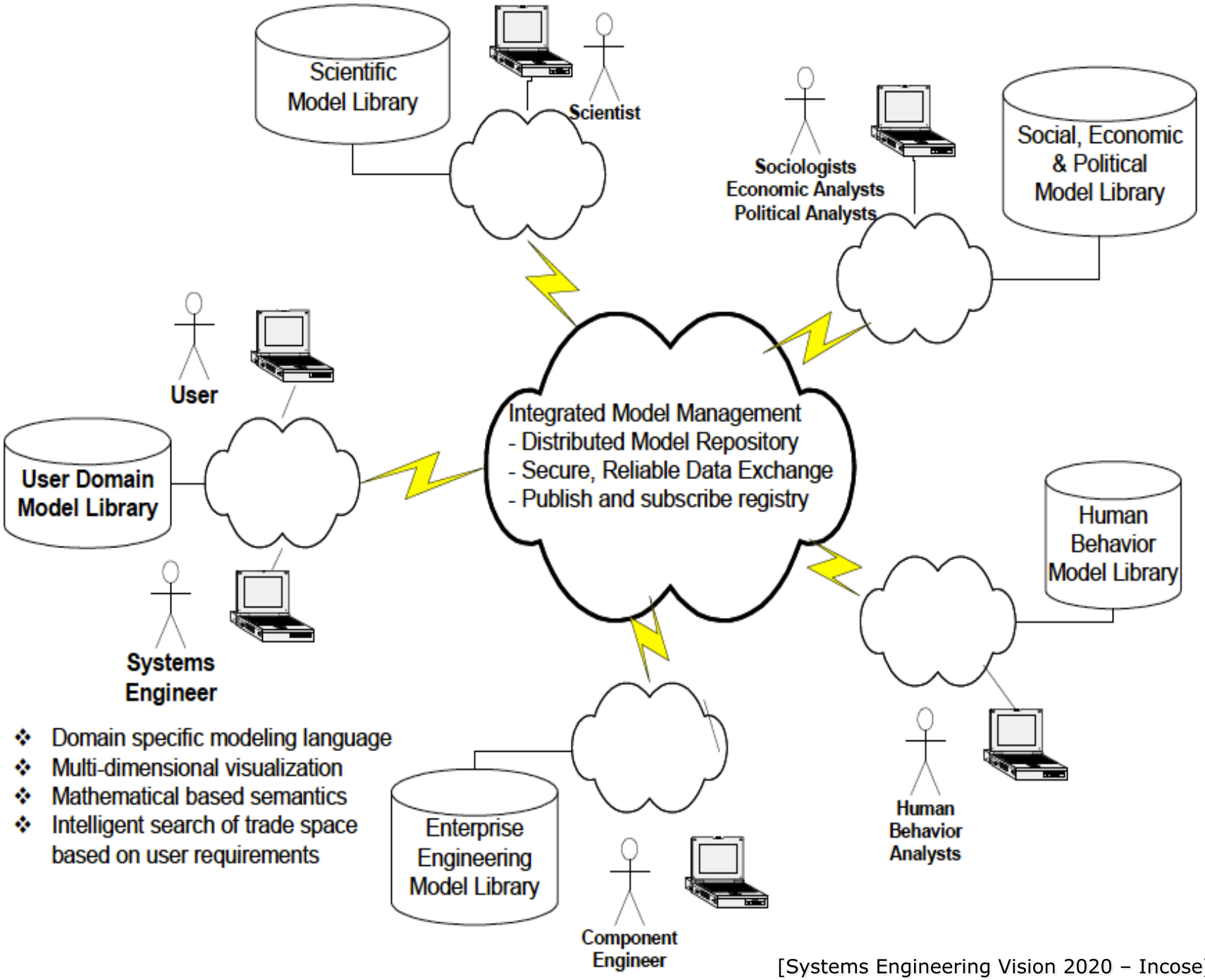
After

Moving from **Document centric**
To **Model centric**

Model Driven System Engineering

- Today
  - « Emerging understanding of SW Eng. and Sys Eng. synergies »
  - « Leaders in the adoption of MBSE are already taking advantage »
- Tomorrow
  - « Future of systems engineering can be said to be model-based »
  - « MBSE is expected to replace the document-centric approach […] and to influence the future practice of Sys. Eng. by being fully integrated into the definition of the Sys. Eng. processes »
  - « DSML enables the systems engineer to focus on modeling of the user domain »

[Systems Engineering Vision 2020 – Incose]



[Systems Engineering Vision 2020 – Incose]

Figure 8 – Cross-domain model integration

UML 1.x une première approche

- Point de départ de la modélisation IS : le *System context diagram*
  - Description des E/S de flots entre le système, les composants, les interfaces, et les éléments de l'environnement.
- *Deployment diagram* seul capable de modéliser les nœuds physiques
  - Mais limité à la «définition d'architectures d'exécution qui représentent l'affectation d'artéfacts logiciels à des nœuds. »
  - Limitatif
- *Object Sequence Diagrams* difficiles à utiliser
  - Pas moyen de les lier à d'autres sous-systèmes
- *Use case* et *state-machine* : répétitifs
- Pas de moyens de lier les diagrammes réalisés aux exigences, ni de modéliser des équations paramétriques

UML 2 : des progrès importants

- Collaboration entre l'IS et l'OMG
- Fournit des moyens de modéliser les structures et les comportement hiérarchiquement
  - *Composite structure diagram*
    - Un seul niveau hiérarchique
    - Pas de modélisation de flots entre sur les liens
- Amélioration des *Objects sequences diagrams*
- Toujours pas de moyens pour :
  - Lier les modèles aux exigences
  - Décrire des flots
  - …

En conclusion : UML et l'IS

- UML bon point de départ
  - Standard de fait dans le monde logiciel
  - UML 2 fournit la plupart des concepts et diagrammes nécessaires pour la description d'un système complexe
  - Mûr et extensible et peut-être adapté pour les exigences de l'IS
  - Nombreux outils disponibles
- Mais ...
  - Manque certains concepts clés de l'IS
  - Utilise des concepts un peu trop « informatique » pour être utilisés par l'ensemble des intervenants du monde IS
  - Trop complet (13 types de diagrammes) pour l'IS

De UML à SysML

- Introduction aux DSL
- De UML à SysML
- Présentation générale de SysML
- Modélisation structurelle
- Modélisation dynamique
- Modélisation transverse
- Réflexion sur SysML
- Conclusion

Objectifs de SysML

- Basé sur UML 2 : support pour la modélisation d'un large panel de sytèmes (hw, sw, données, fabriques…)
- Standard : fournir un langage de modélisation standard pour l'IS pour
  - Analyser, spécifier, concevoir et vérifier les systèmes complexes
  - Permettre l'échange d'informations à travers des outils
  - Franchir le pas sémantique entre les disciplines telles que le système, le logiciel et les autres disciplines de l'IS
- Simple : être facile à apprendre pour des ingénieurs systèmes et facile à implémenter pour des fournisseurs d'outils
- Extensible : pour les domaines particuliers (automobile, aérospatiale …)

Les parties prenantes de SysML

- Industrie
  - American Systems, BAE Systems, Boeing, Deere & Company, EADS Astrium, Eurostep, Israel Aircraft Industries, Lockheed Martin, Motorola, NIST, Northrop Grumman, oose.de, Raytheon, Thales
- Vendeurs d'outils
  - Artisan, EmbeddedPlus, Gentleware, IBM, I-Logix, Mentor Graphics, PivotPoint Technology, Sparx Systems, Telelogic, vitech
- Autres organisations
  - AP-233, INCOSE, Georgia Institute of Technology, AFIS

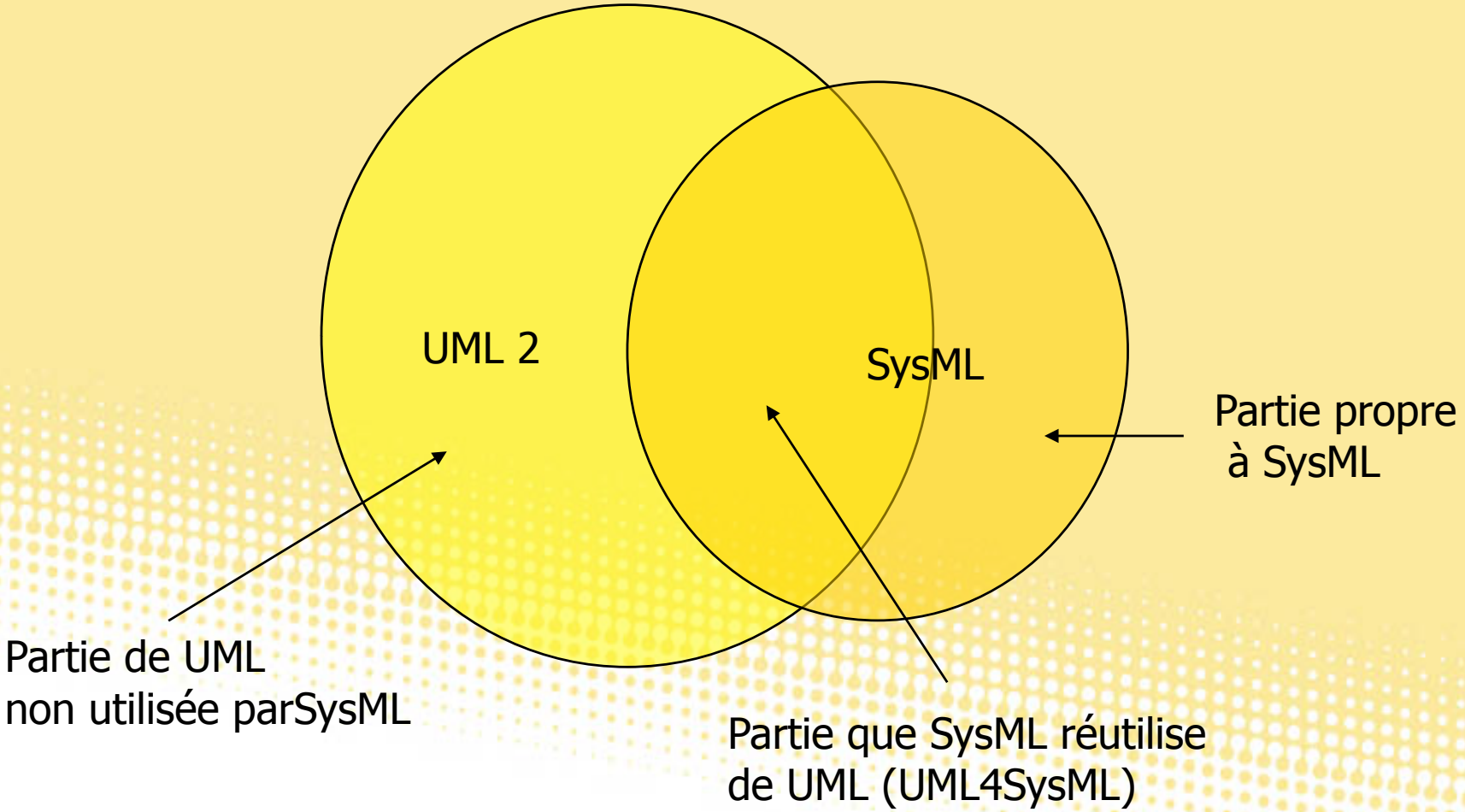
Historique

- UML for System Engineering RFP
  - OMG : mars 2003, sous l'influence de l'INCOSE et AP233
- Initial draft : Janvier 2004
- Séparation de 2 propositions puis regroupement
- SysML Specification v1.0
  - Adoptée par l'OMG en Mai 2006
  - Déclaré « **available specification** » :
    - *September 19, 2007*
  - Document de référence :
    - ptc/07-03-19
  - http://www.omgsysml.org/

UML 2 et l'IS

- Fournit déjà beaucoup de constructions intéressantes pour l'IS
- Décomposition structurelle et interconnexion
  - Parts, Ports, Connectors
- Décomposition comportementale
  - Séquences, activités, états
- Amélioration du diagramme d'activité
  - Flots de données, input/output pin, etx
- Amélioration des diagrammes d'interaction
  - Séquences alternatives, références

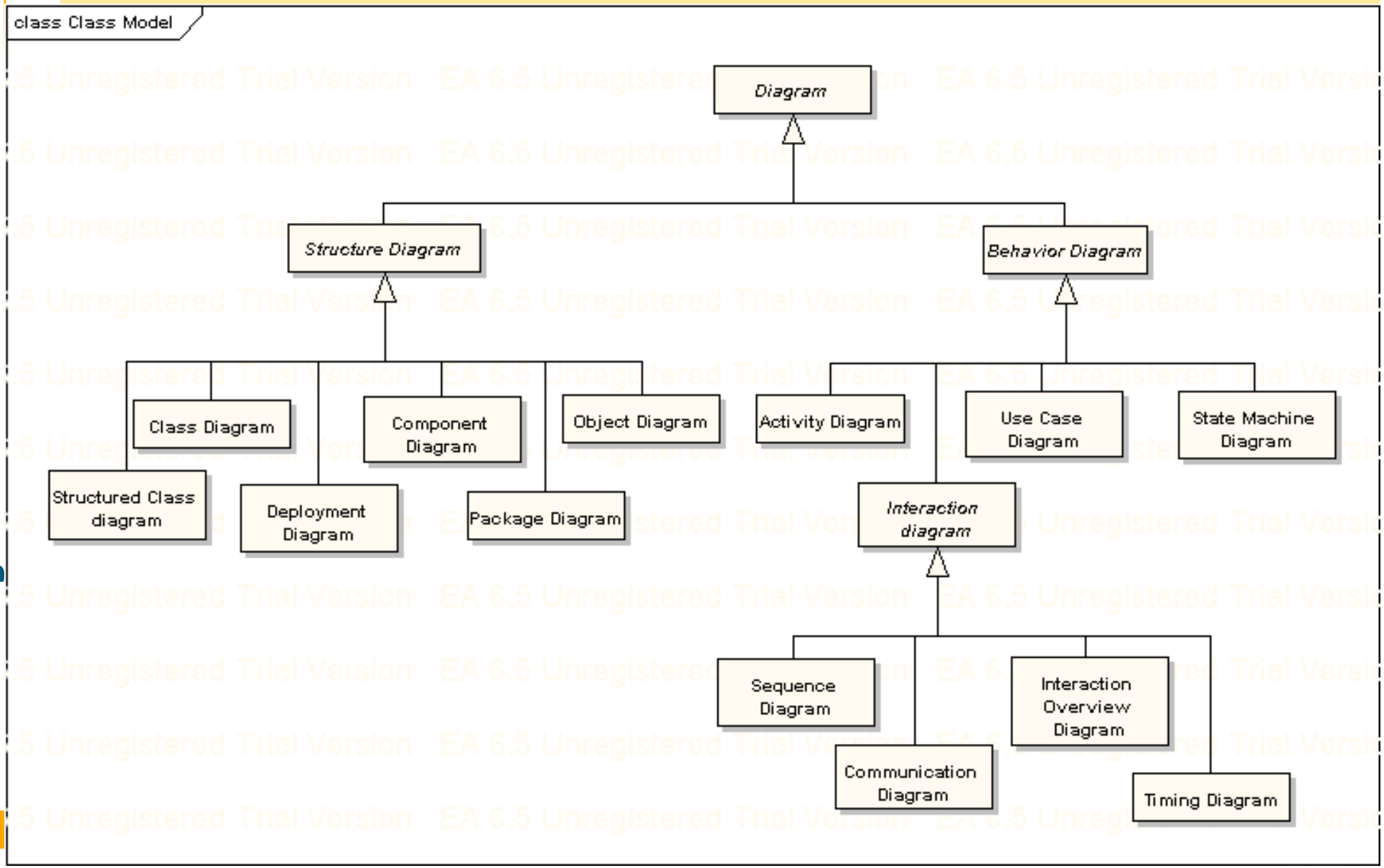
SysML, un profil UML 2



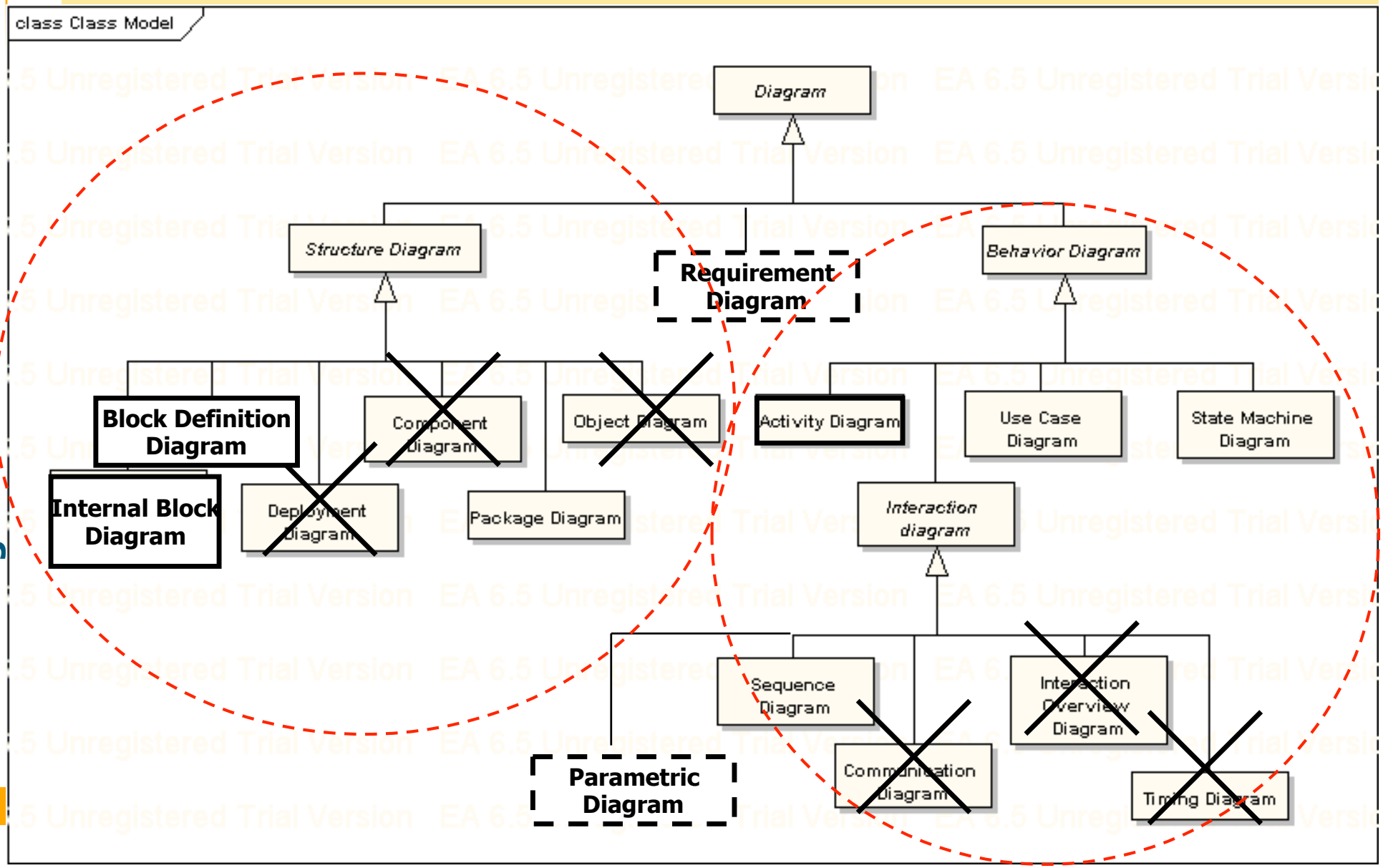
Présentation générale de SysML

- Introduction aux DSL
- De UML à SysML
- Présentation générale de SysML
- Modélisation structurelle
- Modélisation dynamique
- Modélisation transverse
- Réflexion sur SysML
- Conclusion

UML 2 : 13 diagrammes

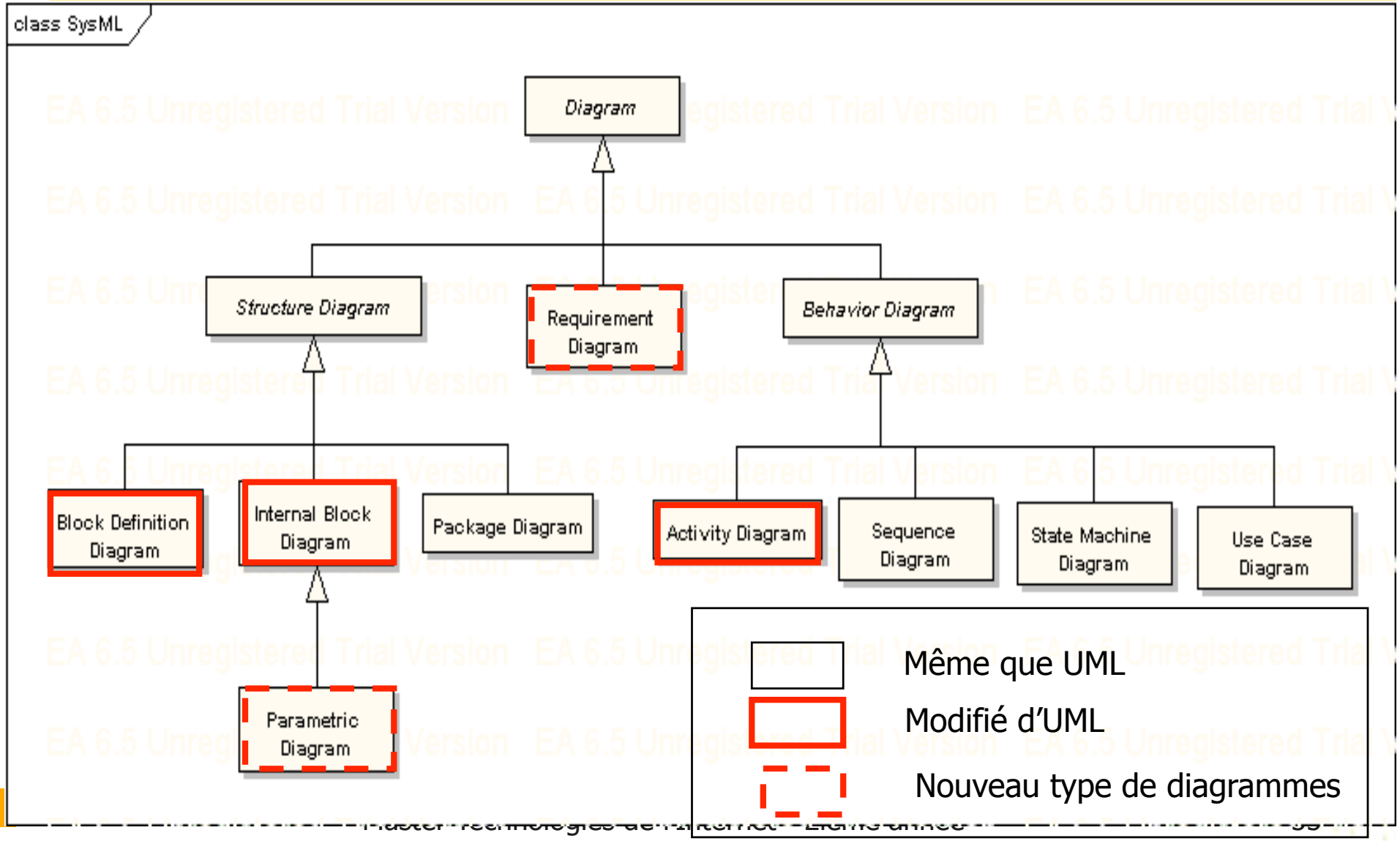


SysML : de 13 à 9 diagrammes



Module Ingénierie des Modèles

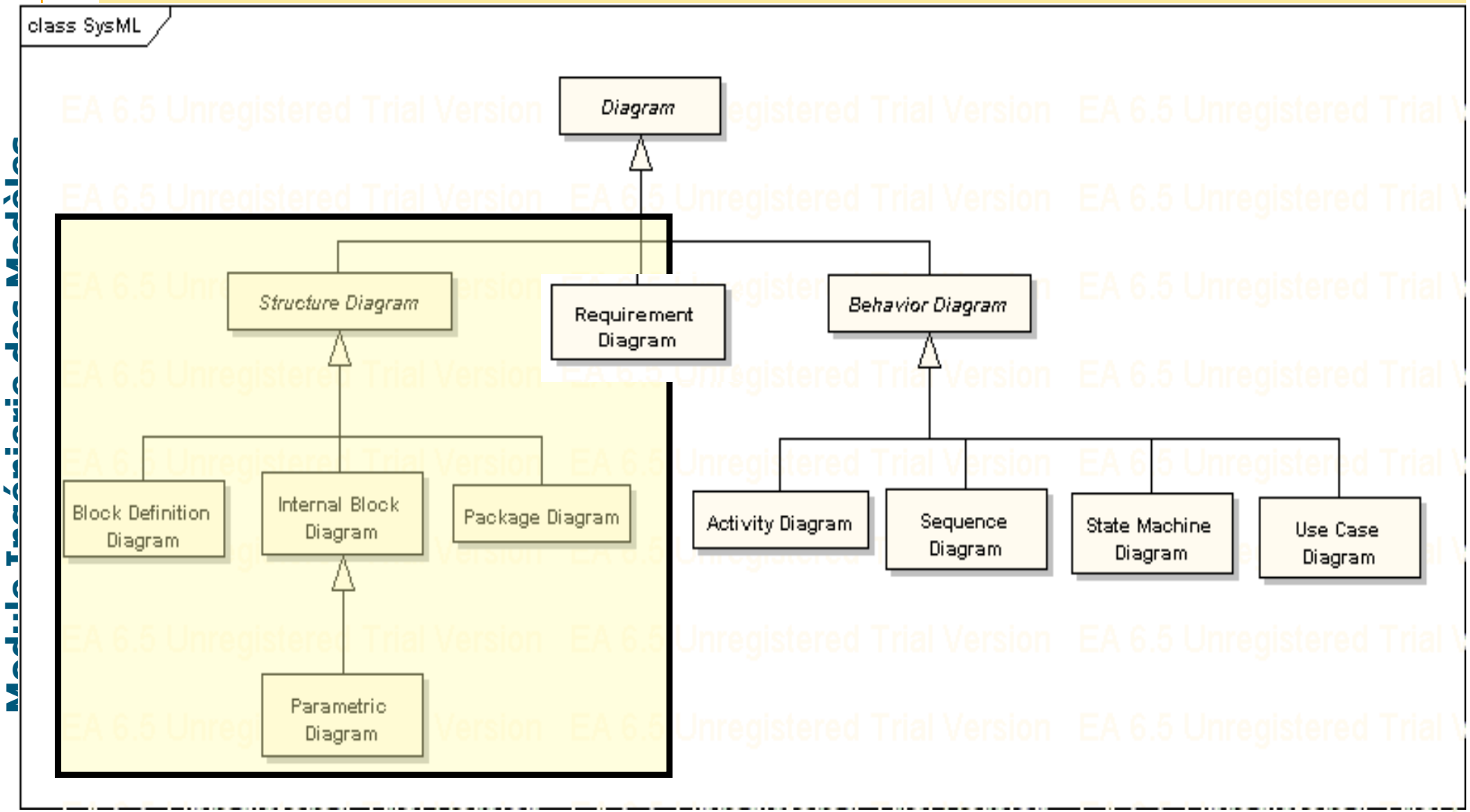
Les diagrammes SysML 1.0



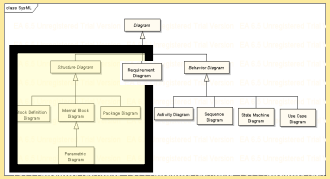
Modélisation structurelle

- Introduction aux DSL
- De UML à SysML
- Présentation générale de SysML
- Modélisation structurelle
- Modélisation dynamique
- Modélisation transverse
- Réflexion sur SysML
- Conclusion

Les diagrammes SysML 1.0

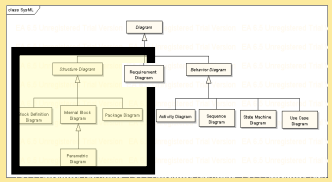


Les packages



- Structuration des modèles
  - Structure arborescente
  - Espaces de noms
  - Fonctionnement collaboratif (espaces de travail
- Organisation variable
  - Par décomposition structurelle
    - Système – sous-systèmes – équipements
  - Par vue ou dommaine
    - Exigences – structure – comportement

Packages



- Exemple

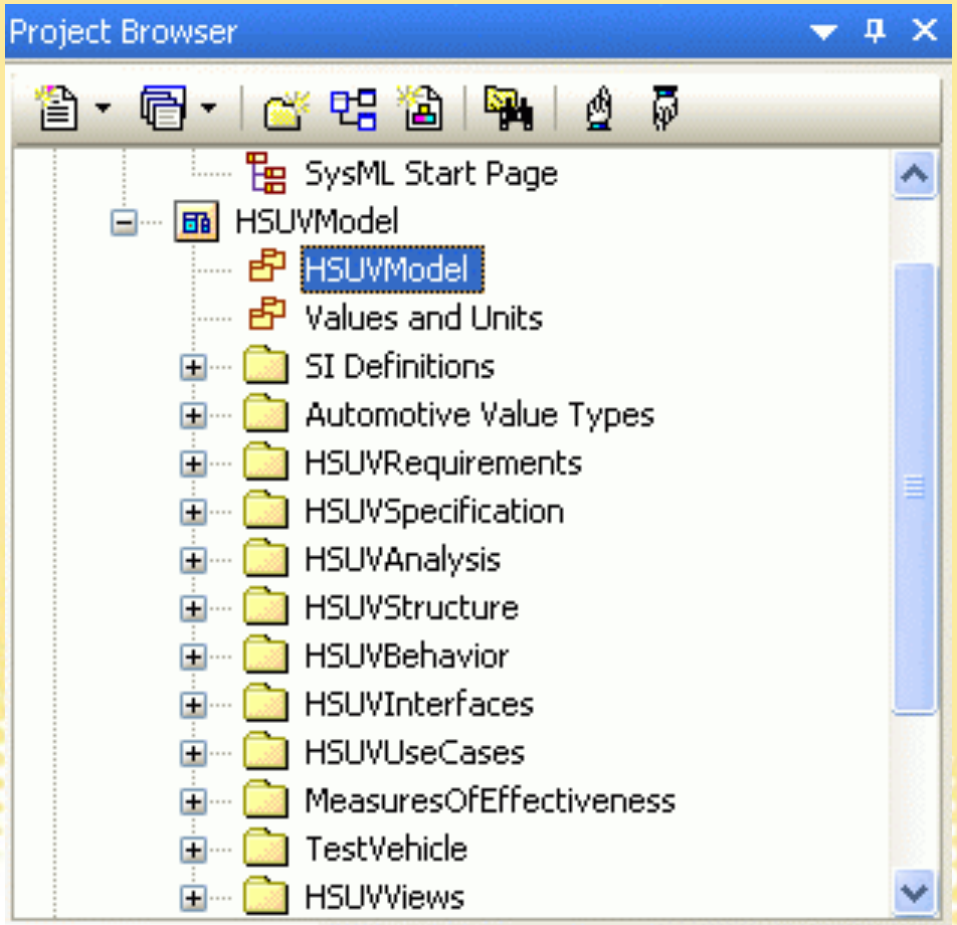
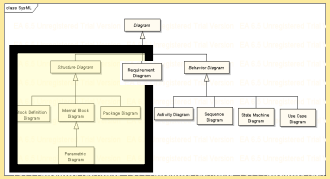
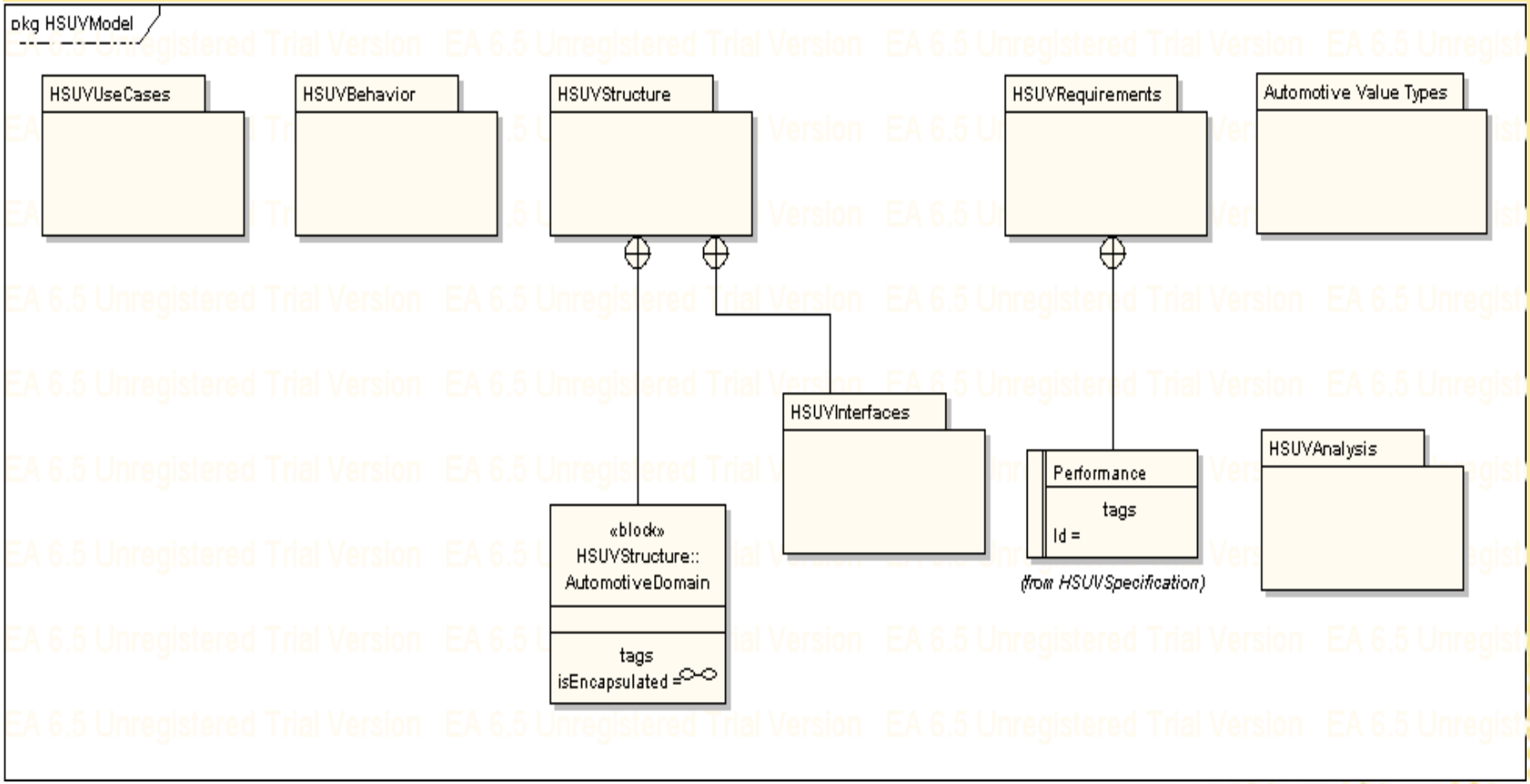


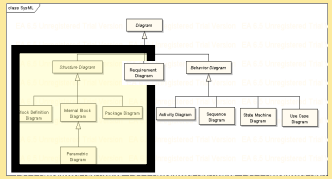
Diagramme de packages



- Liens entre packages

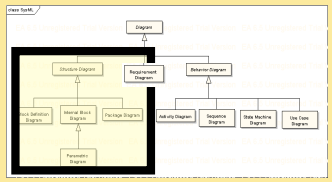


Le concept de *Block*



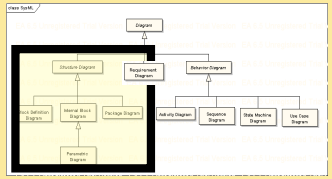
- Concept unifié permettant de décrire la structure d'un élément du système
  - Hardware, software, données, procédure, fabrique, personne, …
- Remplace les concepts UML :
  - Objet/classe, classe structurée, sous-système, composant, nœud, …
- Description des caractéristiques du bloc à travers divers compartiments
  - Propriétés (parts, référence, value), opérations, contraintes, allocations du block (diagrammes d'activité), exigence que le block satisfait

Types de propriétés de block



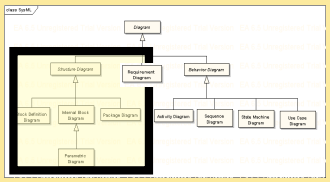
- Une propriété est un élément structurel d'un block
- Propriété *part* (typée par un block)
  - Usage d'un block dans un contexte de vue interne d'un block
  - Exemple : avant-droit:roue
- Propriété référence (typée par un block)
  - Une *part* **non contenue** dans le block courant (pas une composition)
  - Exemple: interface logique entre 2 *parts*
- Propriété valeur (typée par un type de value)
  - Définit une valeur avec ses unités, ses dimensions et ses probabilités de distribution
  - Exemple :
    - Non-distributed value: tirePressure:psi=30
    - Distributed value: «uniform» {min=28,max=32} tirePressure:psi

Utilisation des blocks

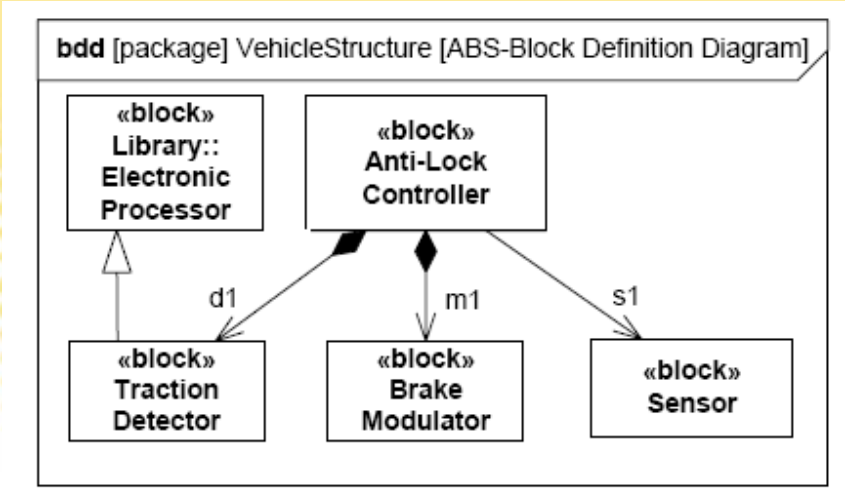


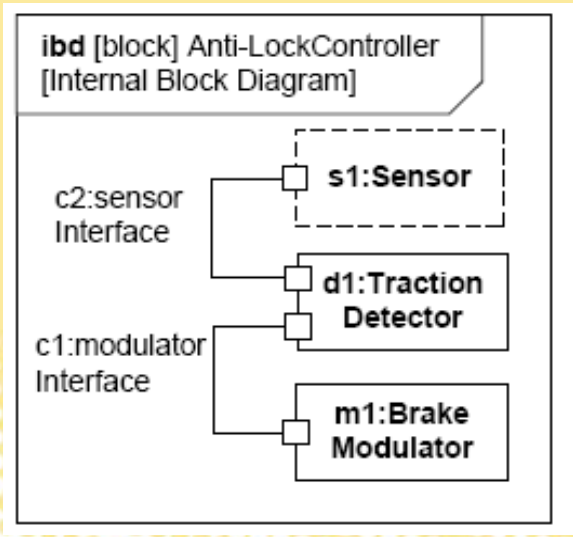
- Basé sur les diagrammes de classe et de classe structurée
  - Élimination des classes d'association
- Les diagrammes de définition de block (BDD) décrivent les relations entre les blocks
  - Association, classification, composition…
- Les diagrammes de block interne (IBD) décrivent la structure interne d'un block en termes de propriétés et de connecteurs
  - On peut associer des comportements à chaque block
- Pour la description des **hiérarchies** et des **interconnexions**

Définition vs. Structure interne

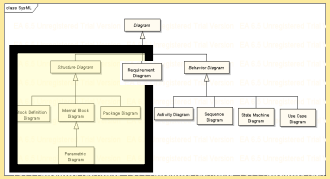


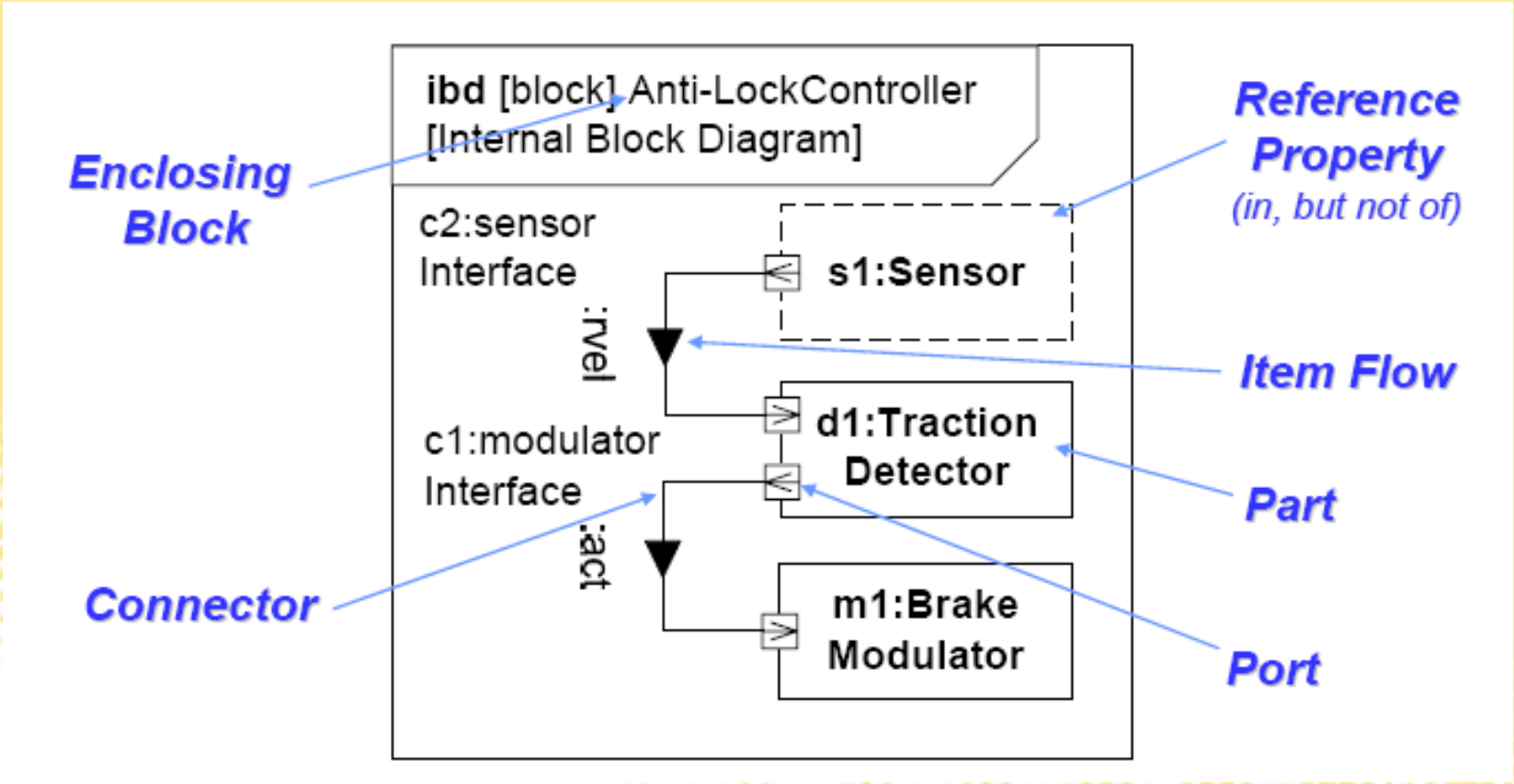
- Diagramme de Définition de blocs (BDD)
  - Décrit les relations entre les blocks (composition, généralisations…)
- Diagramme Interne de bloc (IBD)
  - Décrit la structure interne d'un bloc sous forme de *parts*, *ports* et *connecteurs*.



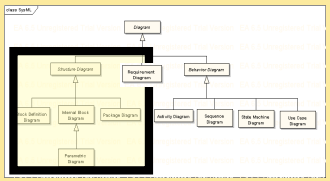


Internal Block Diagram



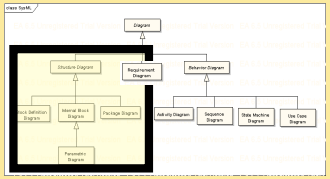


Ports (1/2)

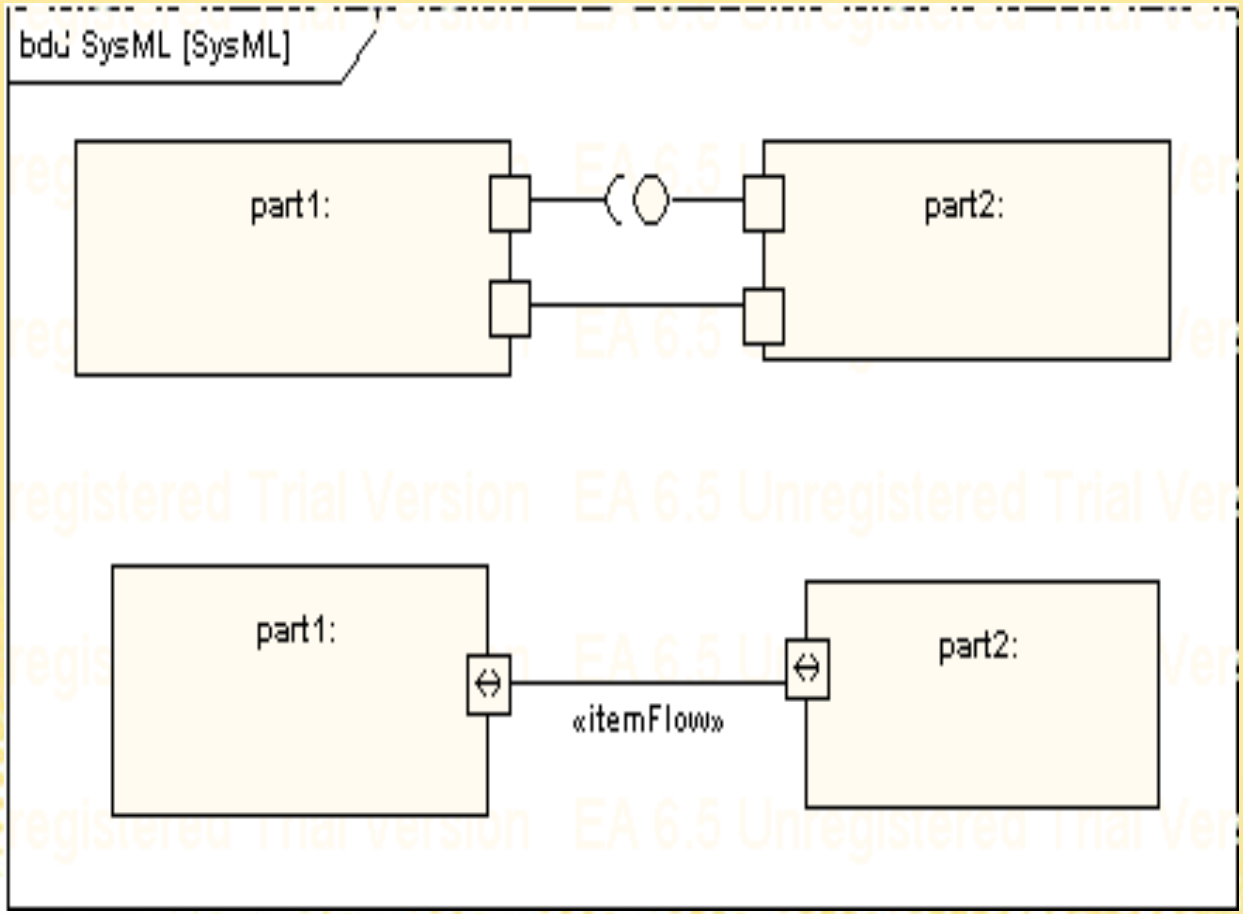


- Spécifient des points d'interactions pour les blocs et les *parts*.
  - Supportent l'intégration de comportements et de structures
- Deux types de ports
  - Standards ports
    - Spécifient un ensemble d'opérations et/ou de signaux
    - Typés par une interface
  - Flow ports
    - Spécifient ce qui circule en entrée ou en sortie d'un bloc ou d'une *part*
    - Typés par une spécification de flot.

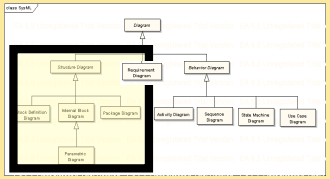
Ports (2/2)

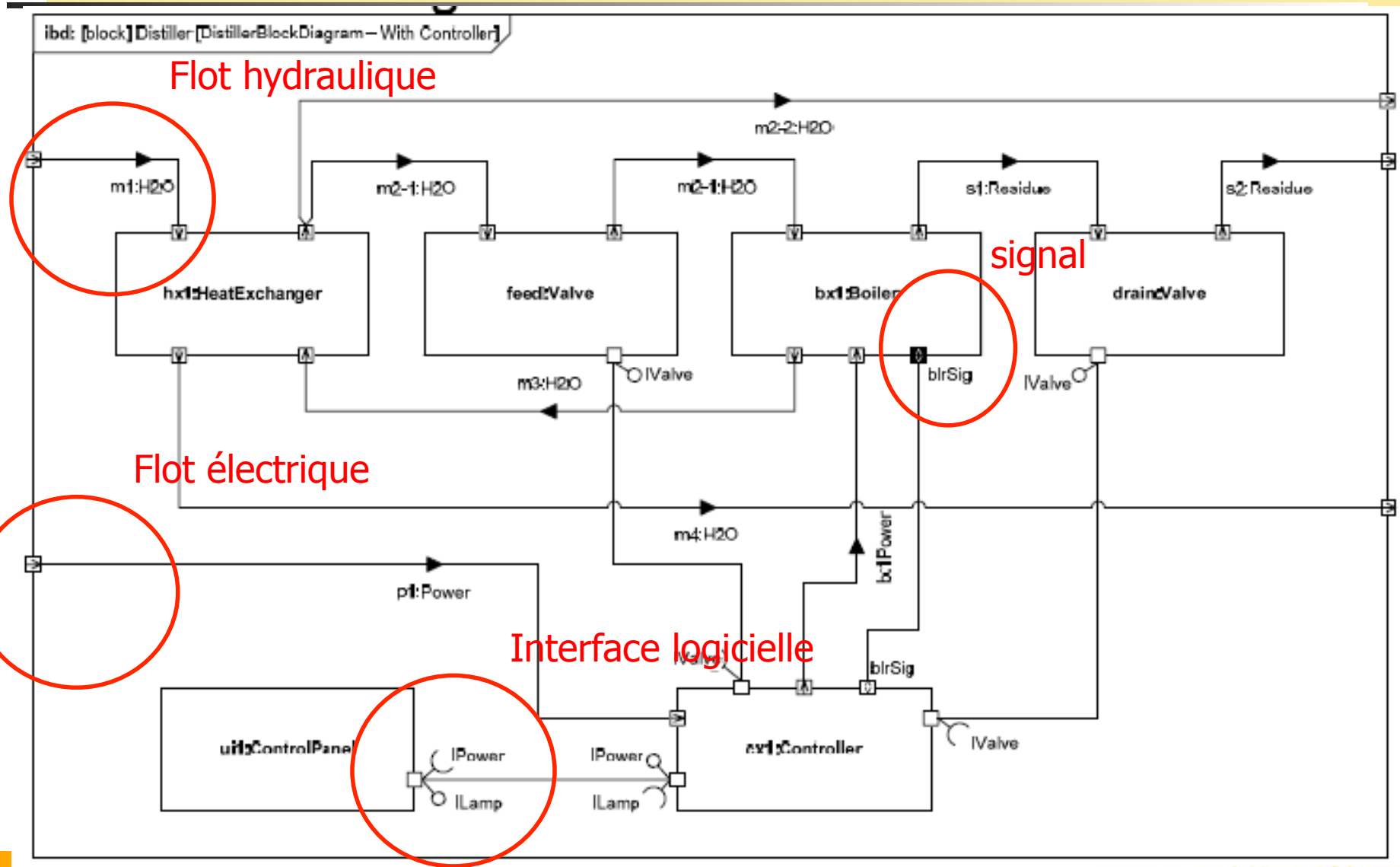


- Standard Port
- Flow Port



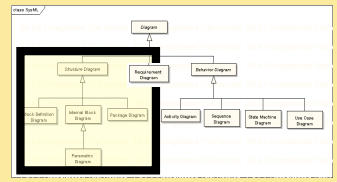
Exemple plus complet





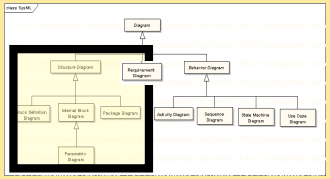
Module Ingénierie des Modèles

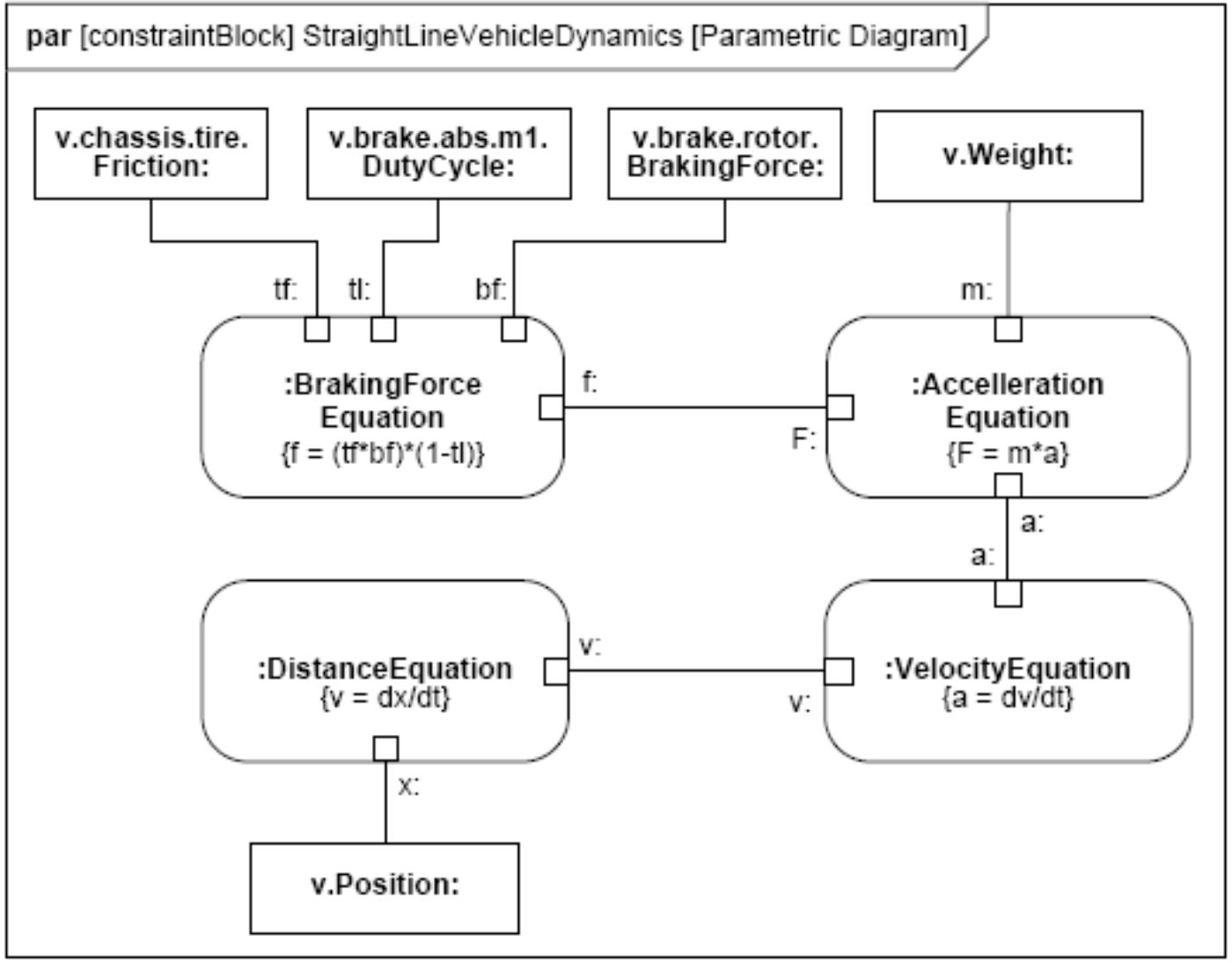
Diagramme paramétrique



- Permet d'exprimer des contraintes (équations) entre les valeurs de paramètres systèmes
  - Performance, fiabilité, masse etc.
- Spécialisation du diagramme de bloc interne
  - Utilisation d'un langage formel ou informel
  - Les seuls blocs utilisables sont des contraintes entre paramètres
  - Permet de représenter graphiquement des équations et relations mathématiques
- Fournit un support pour les études d'analyse système

Exemple





Remarque:
Pas de sens
-> marche
tout le
temps

Modélisation dynamique

- Introduction aux DSL
- De UML à SysML
- Présentation générale de SysML
- Modélisation structurelle
- Modélisation dynamique
- Modélisation transverse
- Réflexion sur SysML
- Conclusion

Modélisation dynamique

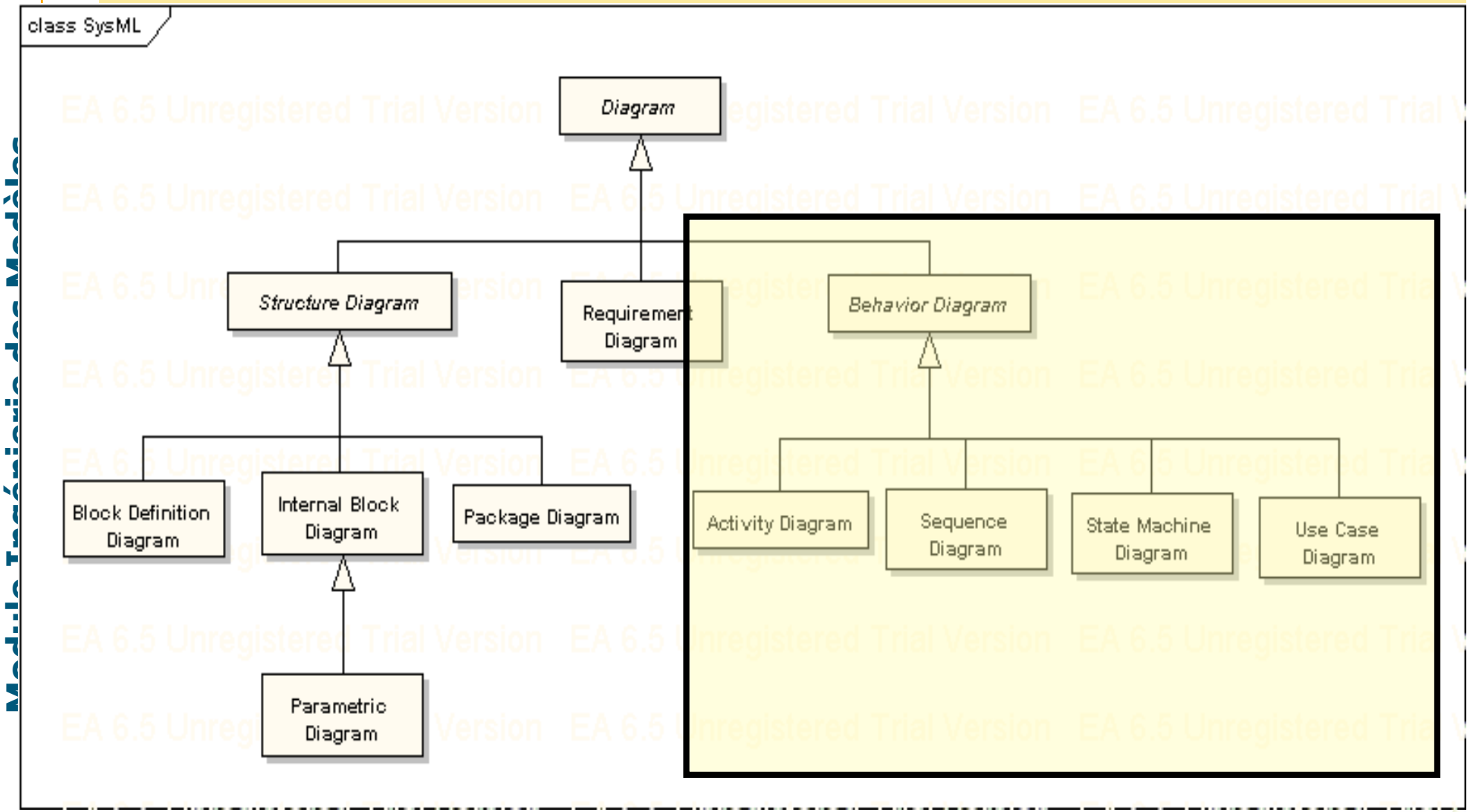
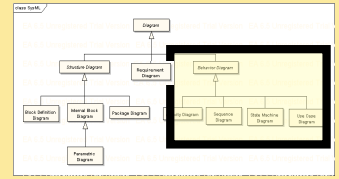
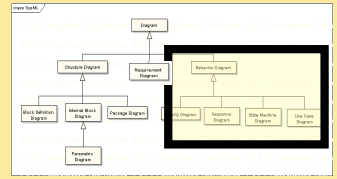


Diagramme de cas d'utilisation



- Idem qu'en UML 2.1
- Peut-être un peu plus d'acteurs non humains?

Diagramme de séquence



- Représentent les éléments intervenant dans un scénario ainsi que les messages et les flots dans un ordre chronologique
- Possibilité de description de scénarios complexes avec les opérateurs *alt*, *loop*, *opt*, *par*,…

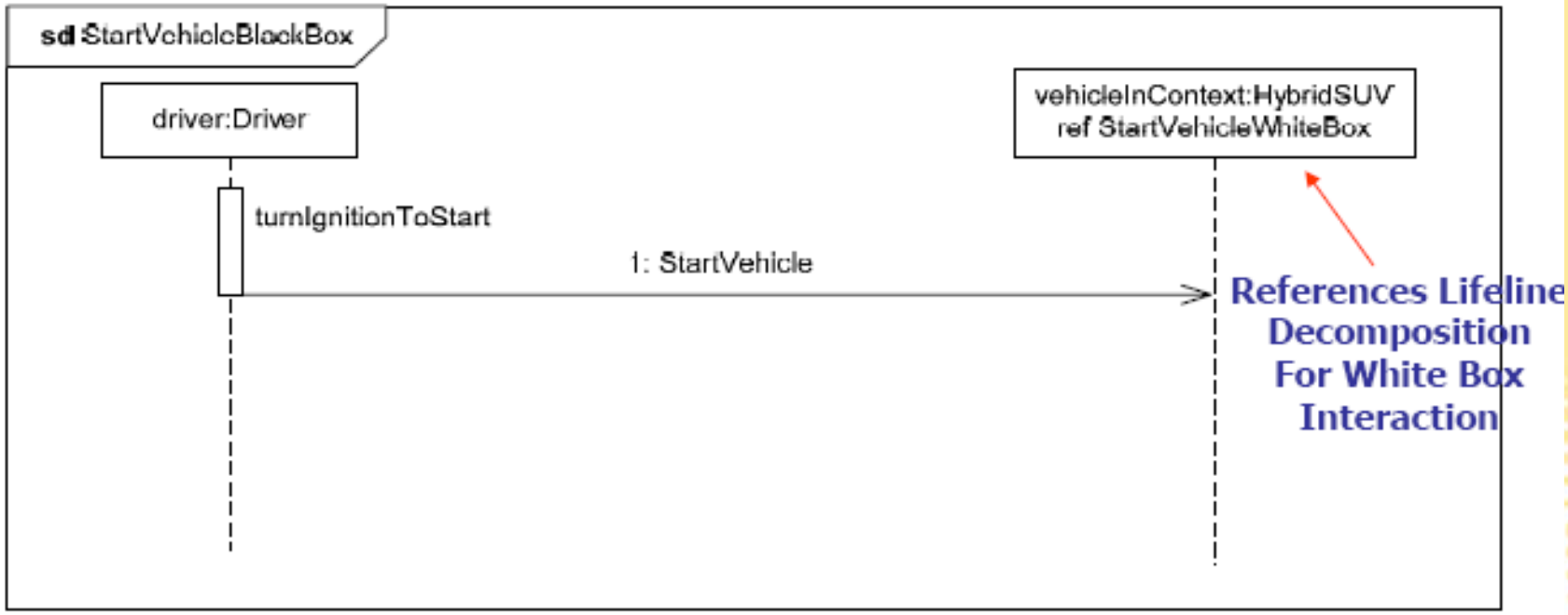
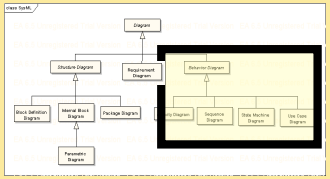


Diagramme de séquence



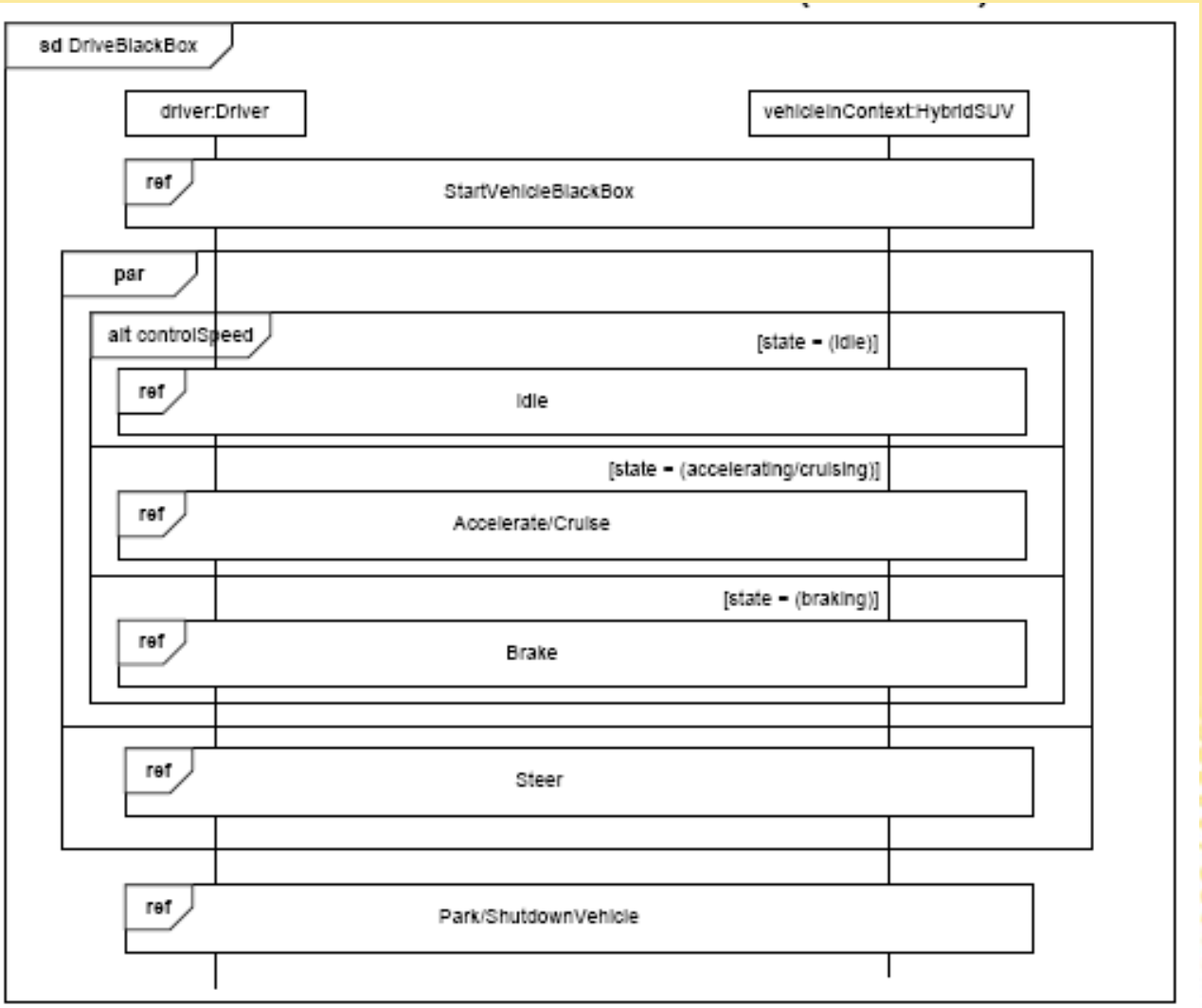
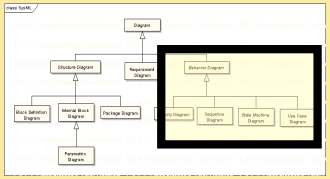
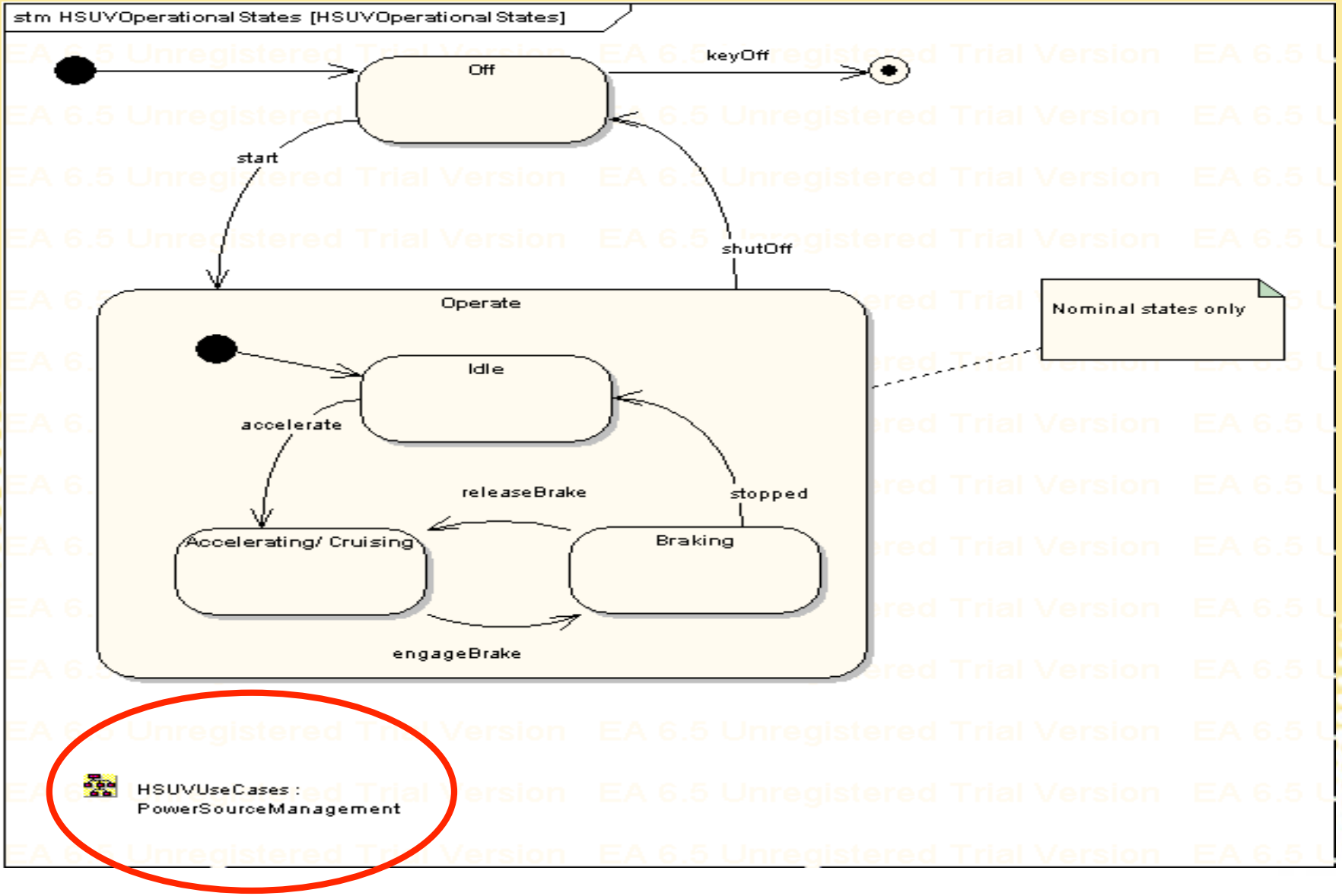
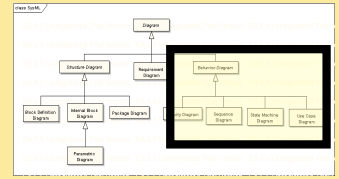


Diagramme d'état/transition





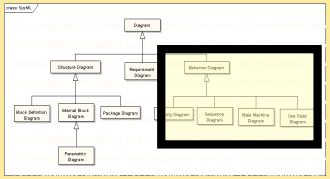
Diagrammes d'activité

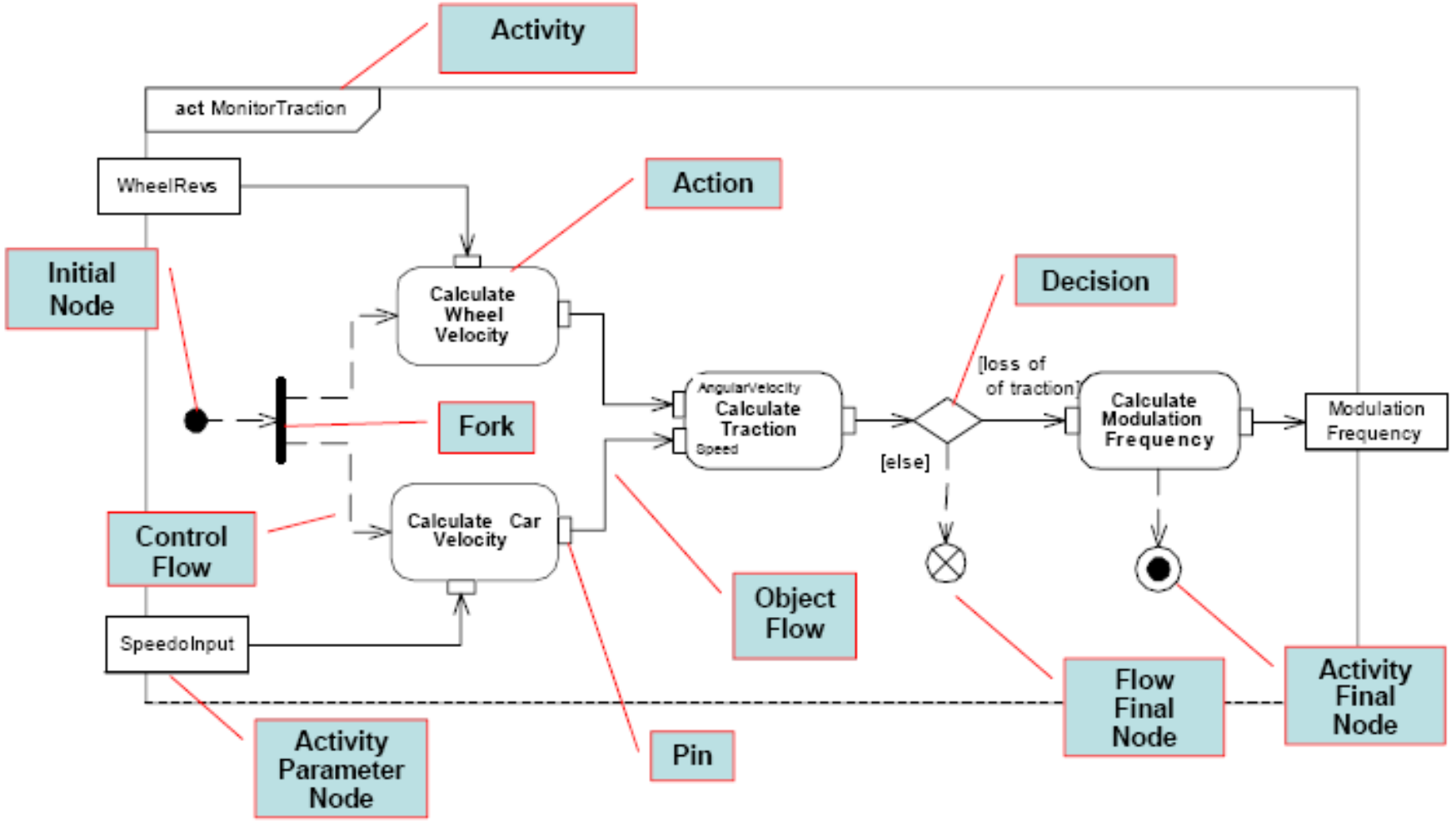


- Une activité est utilisée pour spécifier les flots et les contrôles d'E/S, incluant les séquences et conditions pour les activités coordonnées

- Extensions SysML aux diagrammes d'activités
  - Support pour la modélisation de flots **continus** ou **discrets** (caractérisation de la nature du débit qui circule sur le flot)
  - Introduction d'**opérateurs de contrôles** capables de démarrer ou d'arrêter d'autres actions
  - Introduction des types de ports **Overwrite** et **NoBuffer** pour traiter les flots continus.
  - Possibilité d'assigner des **probabilités** à des arcs ou des ensembles de paramètres.

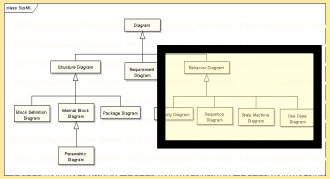
Activités

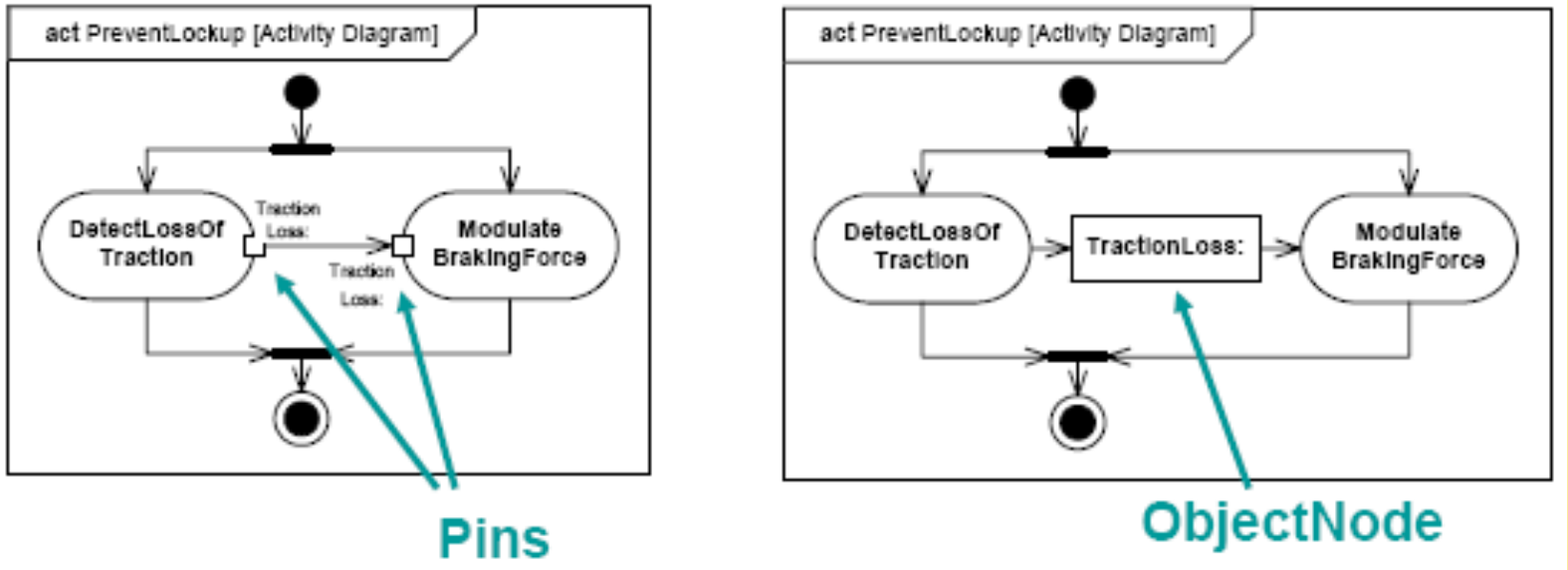




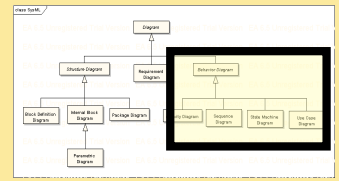
- Join and Merge symbols not included
- Activity Parameter Nodes on frame boundary correspond to activity parameters

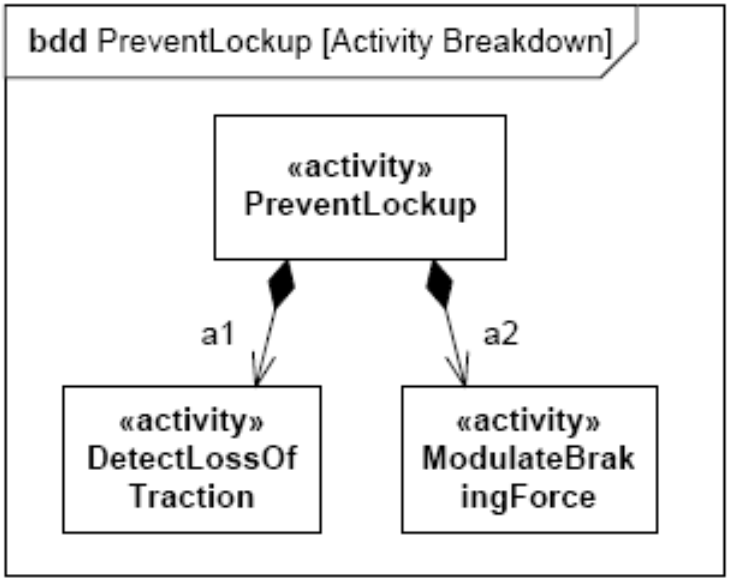
Pins vs ObjetNode

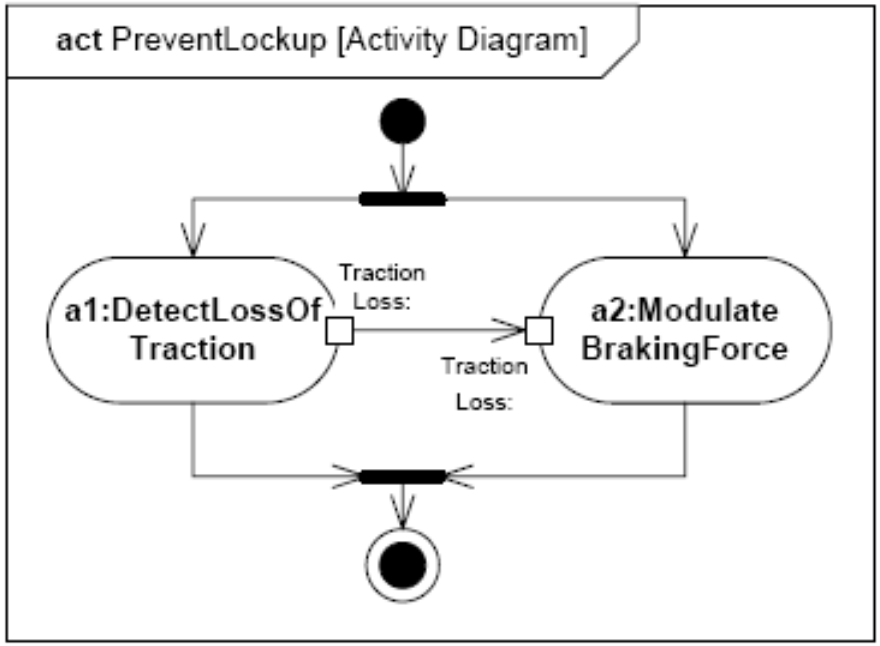




Décomposition d'activité







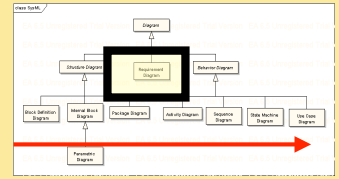
Définition

Utilisation

Modélisation transverse

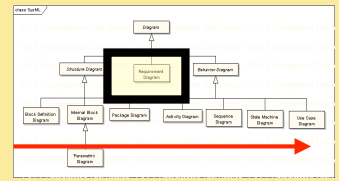
- Introduction aux DSL
- De UML à SysML
- Présentation générale de SysML
- Modélisation structurelle
- Modélisation dynamique
- Modélisation transverse
- Réflexion sur SysML
- Conclusion

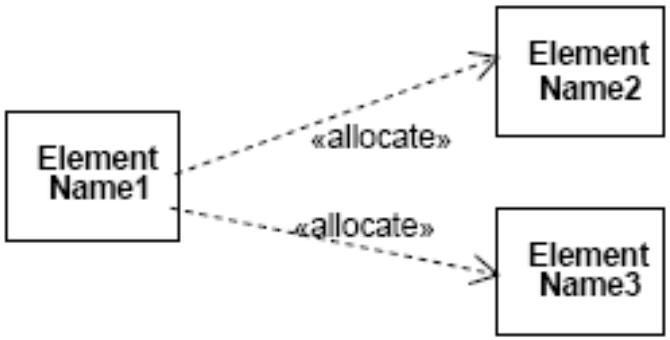
Le concept d'allocation



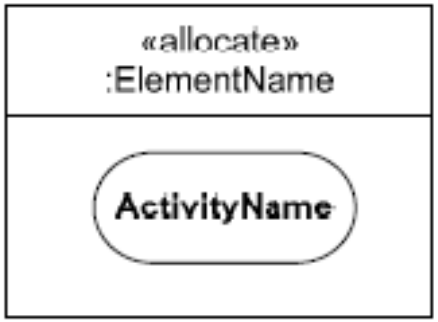
- Relation générale entre un élément d'un modèle et un autre.
- Différents types d'allocation
  - Fonctionnalité - composant
  - Composant logique - composant physique
  - Software – hardware
  - …
- Se retrouve dans de nombreux types de diagramme
- Peuvent également être spécifiées sous forme tabulaire

Exemple de notations

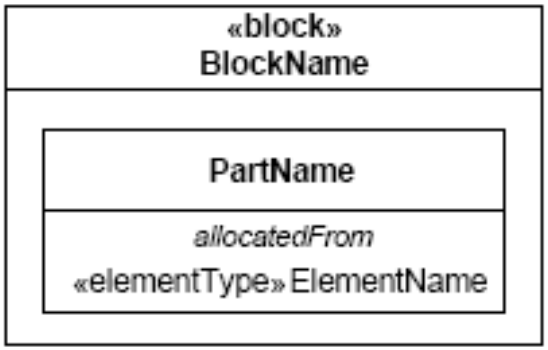




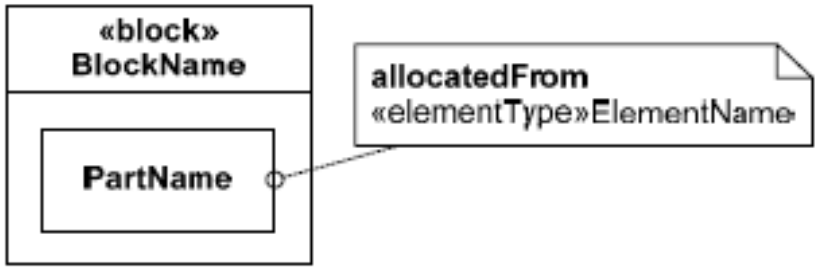
Allocate Relationship



Explicit Allocation of Activity to Swim Lane

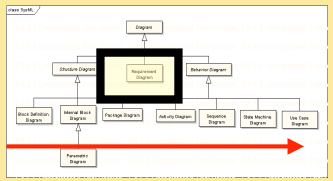


Compartment Notation

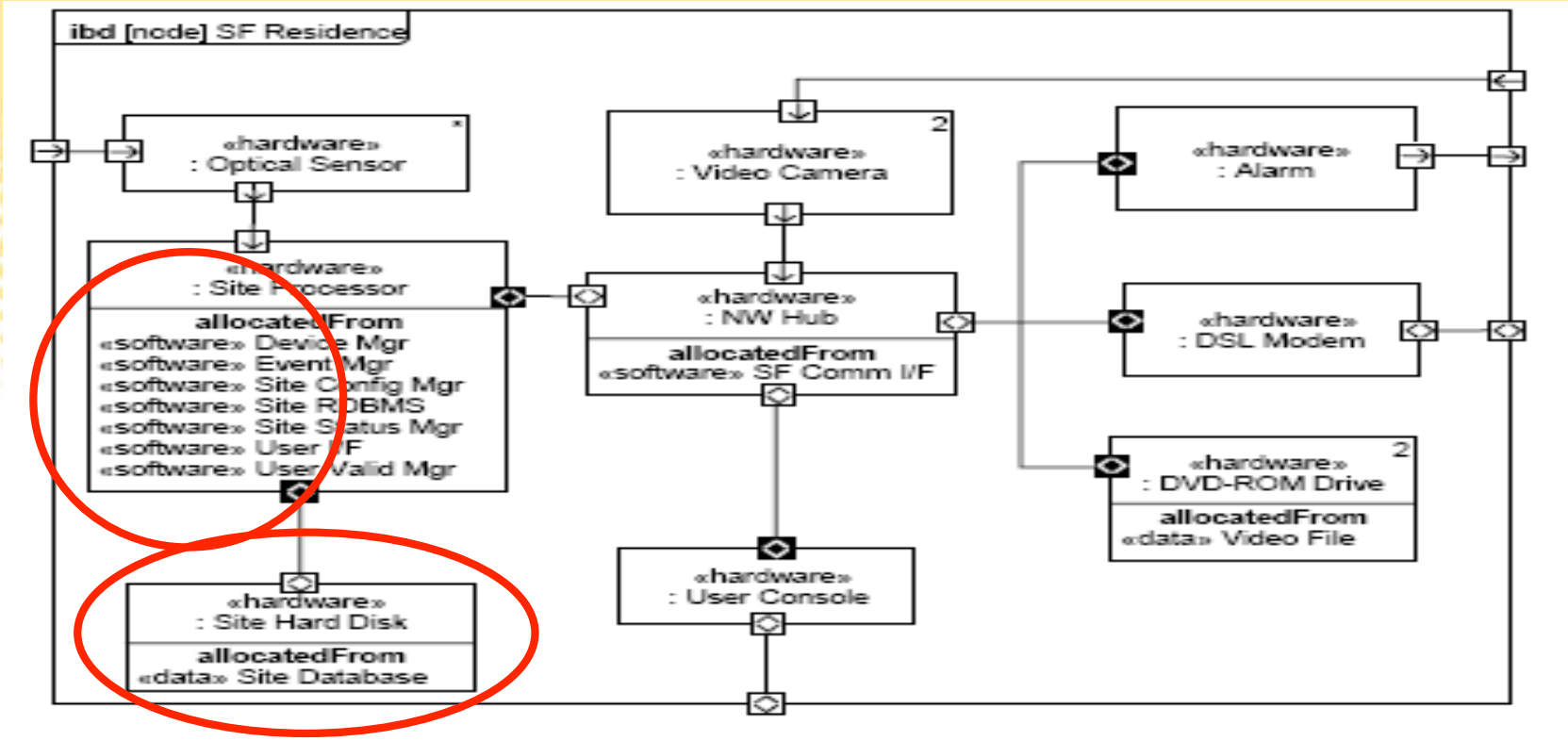


Callout Notation

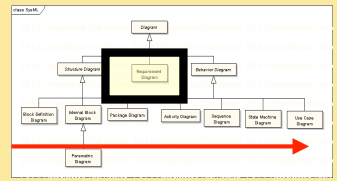
Allocation de software au hardware



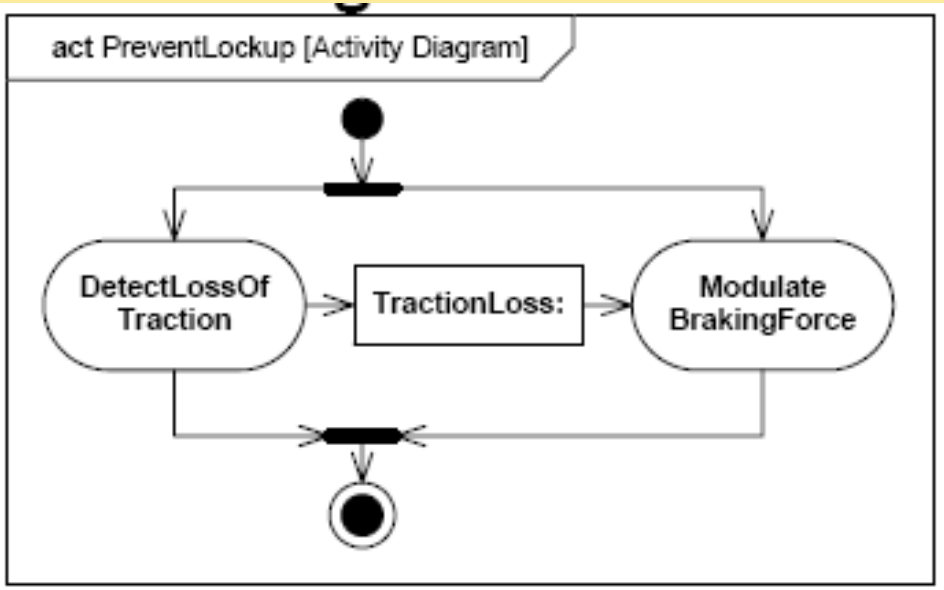
- UML 2
  - le diagramme de déploiement est utilisé pour déployer des artéfacts à des nœuds
- SysML
  - l'allocation sur les IBD et BDD est utilisée pour déployer des éléments logiciels ou des données à des éléments hardware

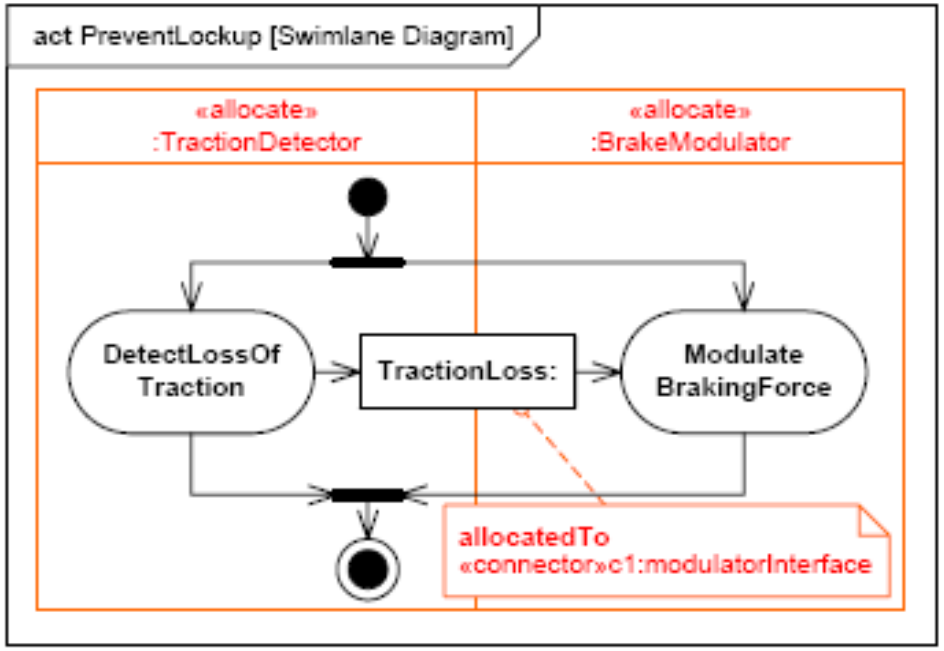


Allocation explicite de comportements

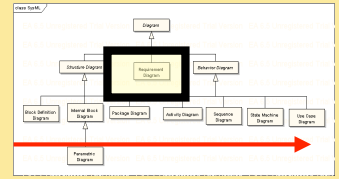


- Utilisation des *swimlanes* (partitions)



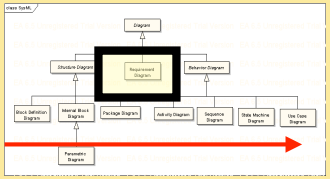


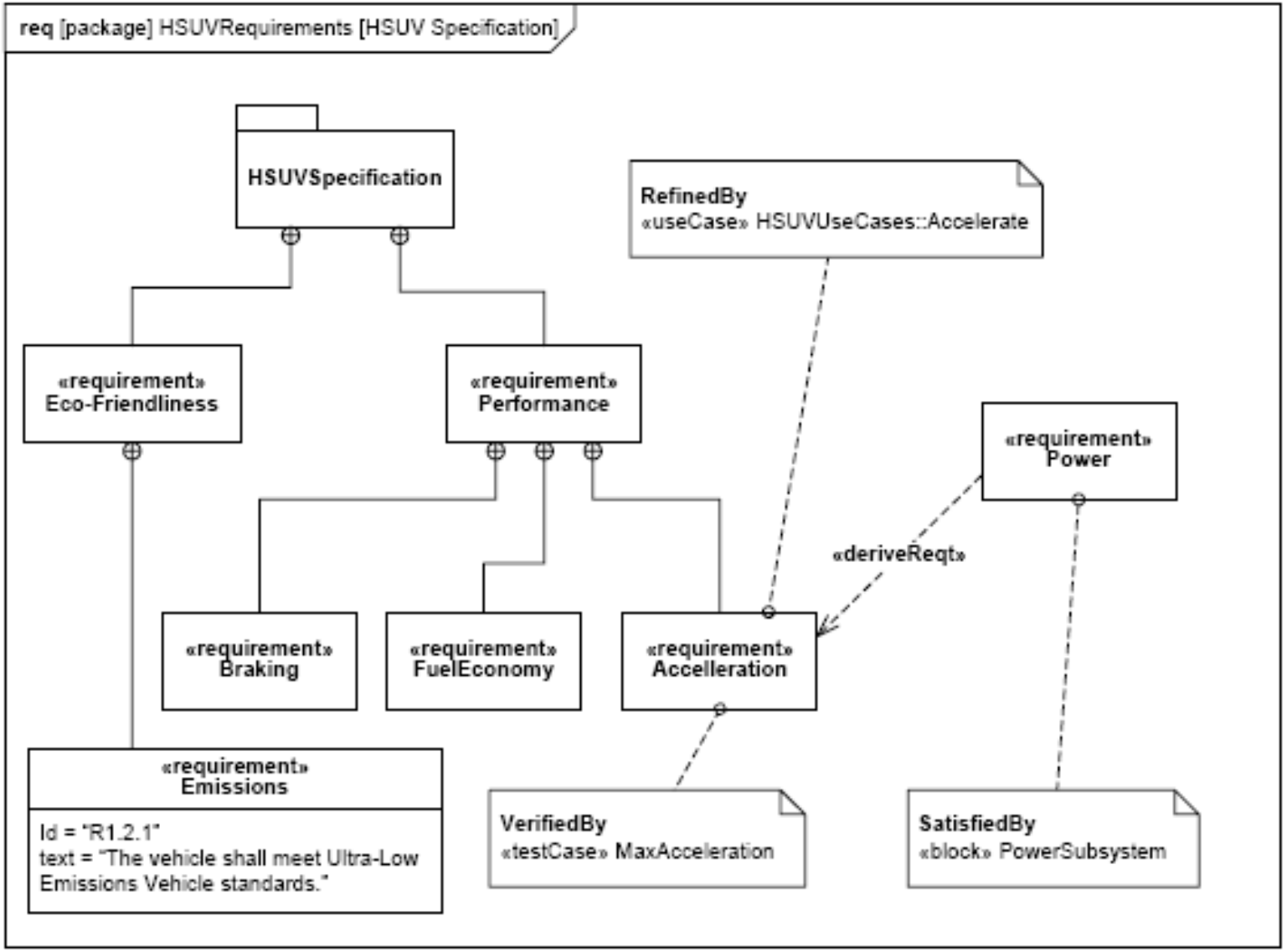
Le diagramme d'exigence



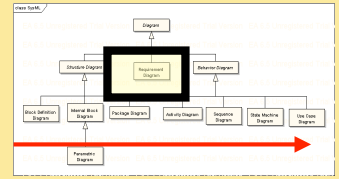
- Stéréotype *requirement* qui permet de représenter des exigences et propriété textuelles
  - Inclut un identifiant *id* et des propriétés textuelles
  - Permet d'ajouter des catégories d'exigence propres à l'utilisateur (i.e. fonctionnelle, interface, performance…)
- Les *requirements* peuvent être décomposées
- Les *requirements* peuvent être spécialisés
- On peut définir des relations entre *requirements* :
  - « deriveRqt »
  - « satisfy »
  - « verify »
  - « refine »
  - « trace »
  - « copy »

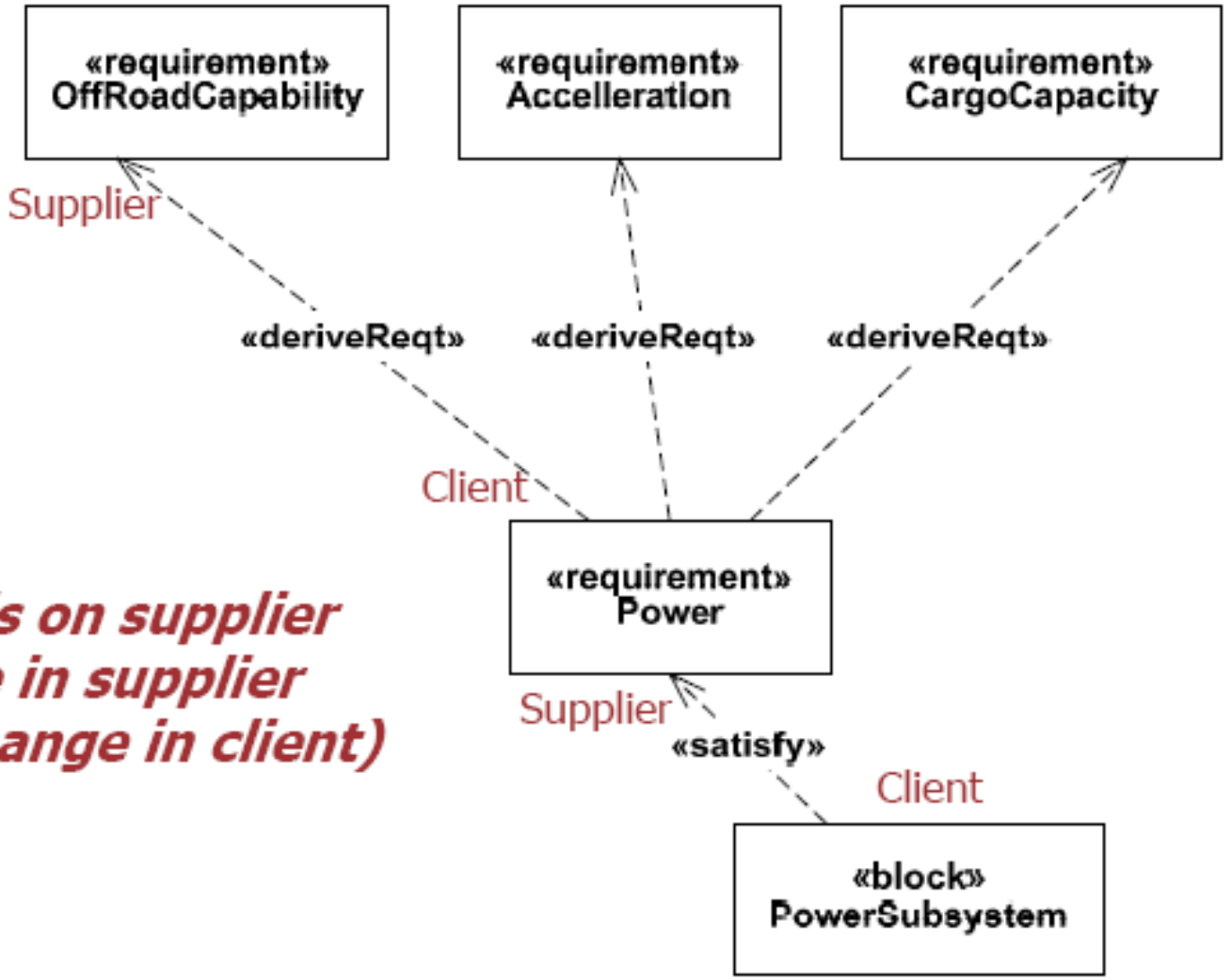
Exemple




req [package] HSUVRequirements [HSUV Specification]
HSUVSpecification
RefinedBy
«useCase» HSUVUseCases::Accelerate
«requirement»
Eco-Friendliness
«requirement»
Performance
«requirement»
Power
«deriveReqt»
«requirement»
Braking
«requirement»
FuelEconomy
«requirement»
Accelleration
«requirement»
Emissions
Id = "R1.2.1"
text = "The vehicle shall meet Ultra-Low Emissions Vehicle standards."
VerifiedBy
«testCase» MaxAcceleration
SatisfiedBy
«block» PowerSubsystem

Exemple de dépendance entre *requirements*





Client depends on supplier (i.e., a change in supplier results in a change in client)

Conclusion

- Introduction aux DSL
- De UML à SysML
- Présentation générale de SysML
- Modélisation structurelle
- Modélisation dynamique
- Modélisation transverse
- Réflexion sur SysML
- Conclusion

SysML, What Else?

- Modeling and Analysis of Real-Time and Embedded Systems (MARTE)
  - A UML profile
  - Real-Time Oriented
- Architecture Analysis and Design Language (AADL)
  - A architecture description language
  - Verification and Validation using tools
  - Extension mechanisms (parser)
  - *http://www.aadl.info/*

Opened Questions 1/

- What maximum granularity level Using SysML?
- How do you translate a SysML model in a UML model?
  - How do you keep links between requirements and corresponding model elements?
  - Model transformation?
- Methodological consideration
  - Which existing method could be adapted within SysML language?

Opened Questions 2/

- SysML genericity : would be a matter?
  - Generic blocks must be stereotyped
- What about the timing concerns?
  - Cooperation/integration with Marte?
- Lack of verification tools for requirement validations

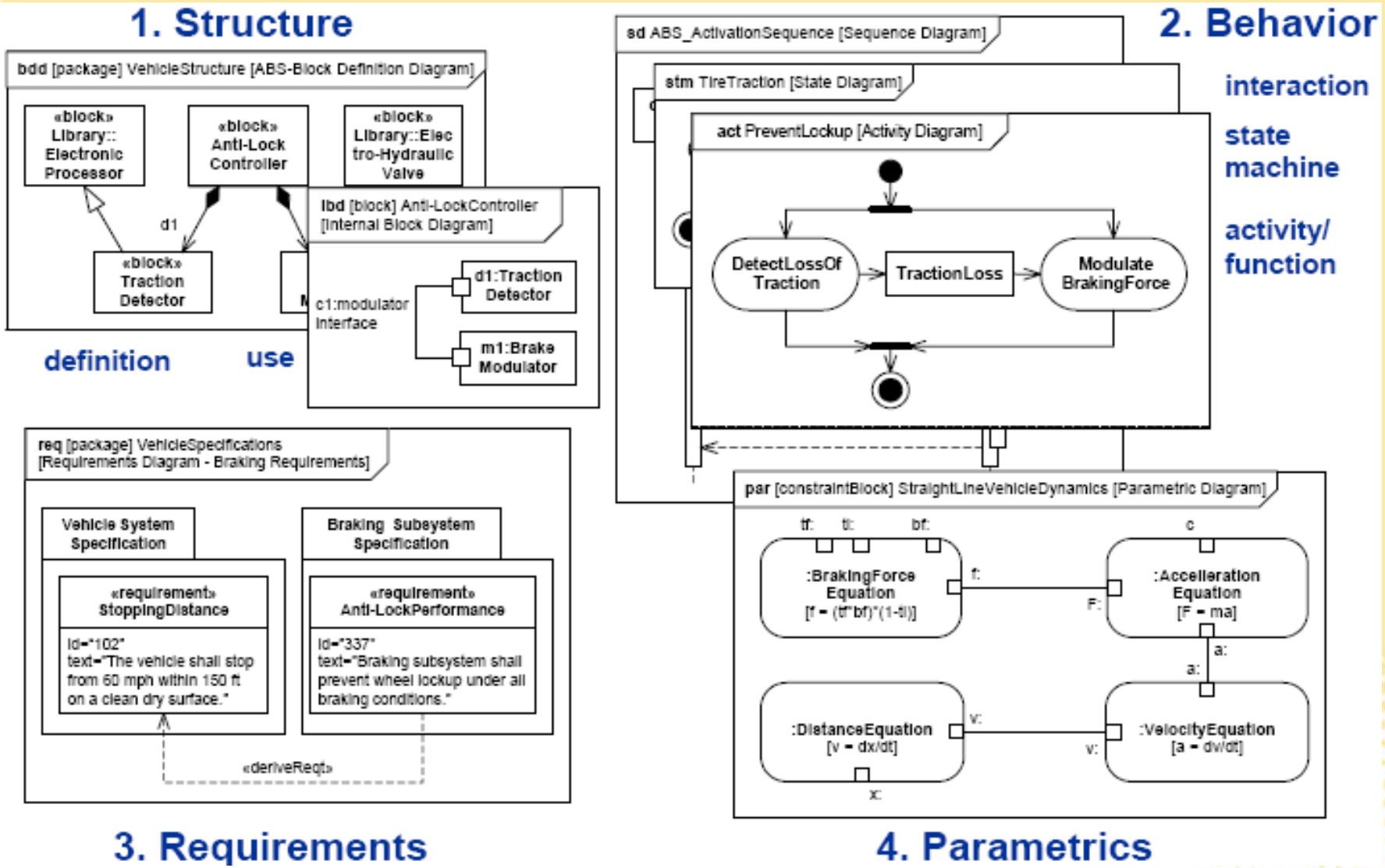
Conclusion

- Introduction aux DSL
- De UML à SysML
- Présentation générale de SysML
- Modélisation structurelle
- Modélisation dynamique
- Modélisation transverse
- Réflexion sur SysML
- Conclusion

Conclusion

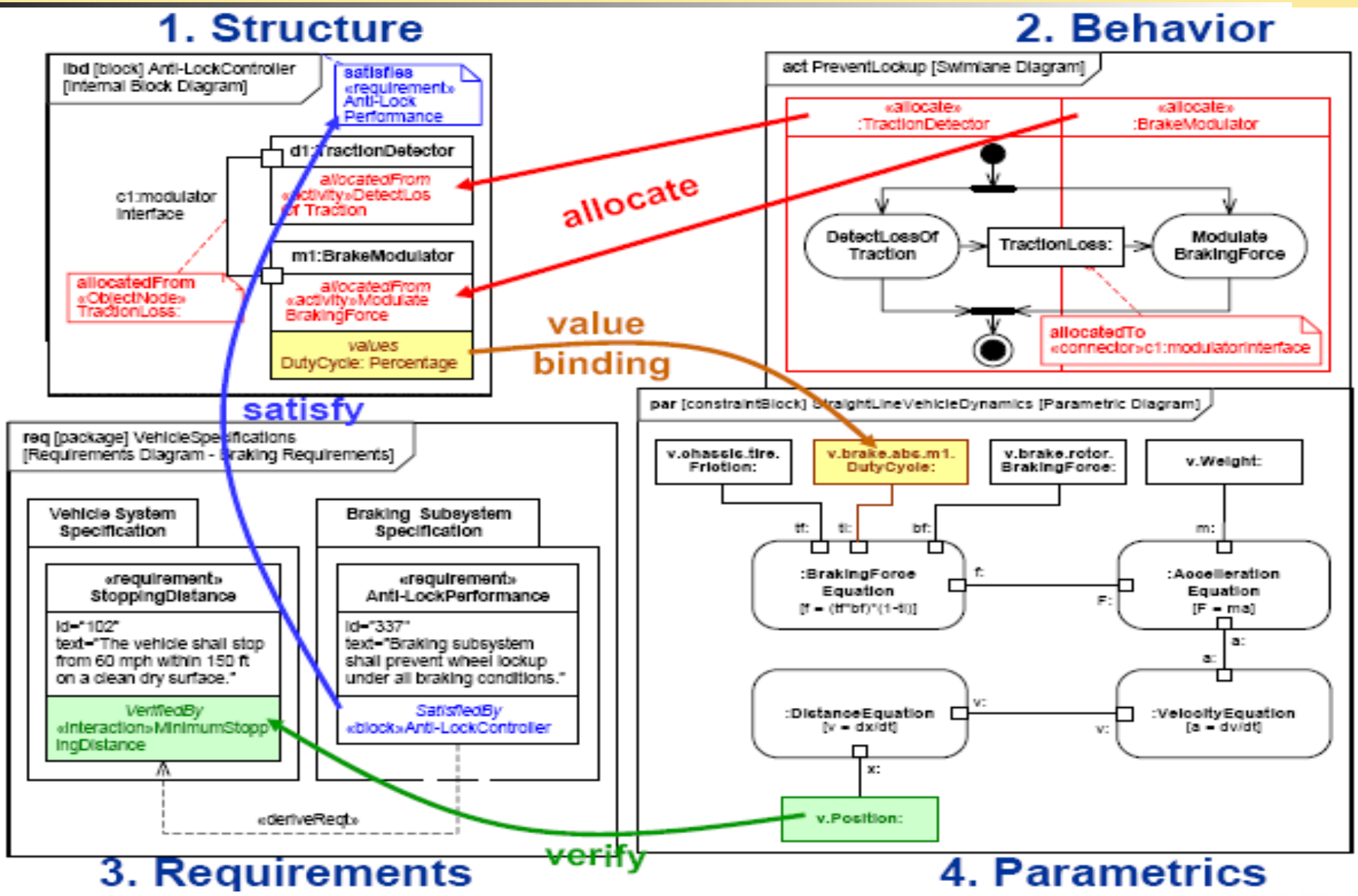
- Un langage spécifique aux systèmes complexes
- Fortement basé UML
- Plutôt axé sur la phase d'analyse
- Avancées notables sur :
  - Liens entre éléments du modèle (exigences, réalisations, allocations…)
  - Modélisation des équations
  - Modélisation des flots continus

Les piliers de SysML



Module Ingénierie des Modèles

Liens entre les éléments



Module Ingénierie des Modèles

Les outils SysML

- Artisan Software / Real-time Studio
  - http://www.artisansw.com/
- Embedded Plus / SysML Toolkit for RSDP
  - http://www.embeddedplus.com/
- I-Logix / Rhapsody
  - http://www.ilogix.com/sublevel.aspx?id=53
- SparxSystems / Enterprise Architect
  - http://www.sparxsystems.com/sysml
- Telelogic / Tau G2
  - http://www.telelogic.com/products/tau/index.cfm
- **Topcased**
  - http://www.topcased.org/

Questions ouvertes

- Comment passer de SysML à UML (et à d'autres langages) ?
  - Conservation des liens entre exigences et diagrammes les réalisant?
  - Transformation de modèles entre les 2?
- Comme en UML, il est nécessaire d'associer une méthode à l'utilisation de SysML
- Généricité de SysML : un problème?
  - Stéréotypage des blocs, une nécessité

Sources

- Sites web
  - http://www.omgsysml.org/
  - http://www.sysml.org/
  - http://www.afis.fr
- Présentations
  - OMG Systems Modeling Language (OMG SysML™) Tutorial
  - Présentation Valtech Training, Pascal Roques
- Documentations
  - « The OMG SysML specifcation v1.0 », OMG parters, ptc-06-05-04
  - « The Systems Modeling Language », Matthew Hause and Alan Moore, ARTiSAN Software Tools, white paper, juin 2006.
  - « An Overview of the Systems Modeling Language for Products and Systems Development », Laurent Balmelli, Oct ' 2006.
  - « Model-driven systems development », L. Balmelli, D. Brown, M. Cantor, M. Mott, July ' 2006.

Des questions?