SysML - Révisions

Synthèse

Lycée Carnot (Dijon), 2020 - 2021

Germain Gondor

SysML : Qu'est-ce ?

Basé sur le langage UML[a], SysML est un langage de modélisation pour l'Ingénierie Système . Il prend en charge la spécification, l'analyse, la conception, la vérification et la validation des systèmes qui comprennent le matériel, les logiciels, les données, le personnel, les procédures et les installations.

SysML : Qu'est-ce ?

Basé sur le langage UML[a], SysML est un langage de modélisation pour l'Ingénierie Système . Il prend en charge la spécification, l'analyse, la conception, la vérification et la validation des systèmes qui comprennent le matériel, les logiciels, les données, le personnel, les procédures et les installations.

C'est un langage de modélisation qui fournit :

SysML : Qu'est-ce ?

Basé sur le langage UML[a], SysML est un langage de modélisation pour l'Ingénierie Système . Il prend en charge la spécification, l'analyse, la conception, la vérification et la validation des systèmes qui comprennent le matériel, les logiciels, les données, le personnel, les procédures et les installations.

C'est un langage de modélisation qui fournit :

- **une sémantique :** qui donne une signification et une relation entre les signes et leurs référents

SysML : Qu'est-ce ?

Basé sur le langage UML[a], SysML est un langage de modélisation pour l'Ingénierie Système . Il prend en charge la spécification, l'analyse, la conception, la vérification et la validation des systèmes qui comprennent le matériel, les logiciels, les données, le personnel, les procédures et les installations.

C'est un langage de modélisation qui fournit :

- **une sémantique :** qui donne une signification et une relation entre les signes et leurs référents
- **une notation :** qui est un ensemble de signes conventionnels qui servent à fixer par écrit leur interprétation.

SysML : Qu'est-ce ?

Basé sur le langage UML[a], SysML est un langage de modélisation pour l'Ingénierie Système . Il prend en charge la spécification, l'analyse, la conception, la vérification et la validation des systèmes qui comprennent le matériel, les logiciels, les données, le personnel, les procédures et les installations.

C'est un langage de modélisation qui fournit :

- **une sémantique :** qui donne une signification et une relation entre les signes et leurs référents
- **une notation :** qui est un ensemble de signes conventionnels qui servent à fixer par écrit leur interprétation.

SysML permet d'approcher un modèle par des vues (fenêtre ayant un angle de vision déterminé). Une vue est un élément du modèle. Trois points de vue ont été privilégiés dans le langage SysML.

a. Unified Modeling Language

- Un point de vue comportemental, auxquels sont associés quatre diagrammes :

- Un point de vue comportemental, auxquels sont associés quatre diagrammes :
  - Le diagramme des cas d'utilisation (*Use Case Diagram*, indicateur **uc** ou **ucd**)

- Un point de vue comportemental, auxquels sont associés quatre diagrammes :
  - Le diagramme des cas d'utilisation (*Use Case Diagram*, indicateur **uc** ou **ucd**)
  - Le diagramme de séquence (*Sequence Diagram*, indicateur **seq**)

- Un point de vue comportemental, auxquels sont associés quatre diagrammes :
  - Le diagramme des cas d'utilisation (*Use Case Diagram*, indicateur **uc** ou **ucd**)
  - Le diagramme de séquence (*Sequence Diagram*, indicateur **seq**)
  - Le diagramme d'états (*State Machine Diagram*, indicateur **stm**)

- Un point de vue comportemental, auxquels sont associés quatre diagrammes :
  - Le diagramme des cas d'utilisation (*Use Case Diagram*, indicateur **uc** ou **ucd**)
  - Le diagramme de séquence (*Sequence Diagram*, indicateur **seq**)
  - Le diagramme d'états (*State Machine Diagram*, indicateur **stm**)
  - Le diagramme d'activités (*Activity Diagram*, indicateur **act**)

- Un point de vue comportemental, auxquels sont associés quatre diagrammes :
  - Le diagramme des cas d'utilisation (*Use Case Diagram*, indicateur **uc** ou **ucd**)
  - Le diagramme de séquence (*Sequence Diagram*, indicateur **seq**)
  - Le diagramme d'états (*State Machine Diagram*, indicateur **stm**)
  - Le diagramme d'activités (*Activity Diagram*, indicateur **act**)
- Un point de vue structurel, auxquels sont associés quatre diagrammes :

- Un point de vue comportemental, auxquels sont associés quatre diagrammes :
  - Le diagramme des cas d'utilisation (*Use Case Diagram*, indicateur **uc** ou **ucd**)
  - Le diagramme de séquence (*Sequence Diagram*, indicateur **seq**)
  - Le diagramme d'états (*State Machine Diagram*, indicateur **stm**)
  - Le diagramme d'activités (*Activity Diagram*, indicateur **act**)
- Un point de vue structurel, auxquels sont associés quatre diagrammes :
  - Le diagramme de définition de blocs (*Block Definition Diagram*, indicateur **bdd**)

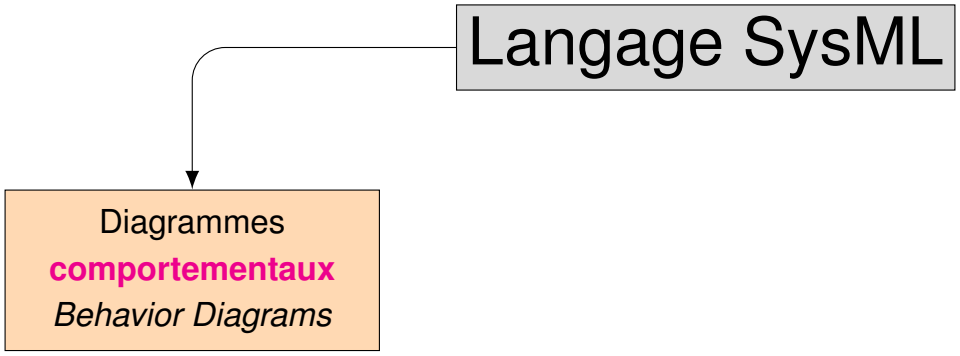
- Un point de vue comportemental, auxquels sont associés quatre diagrammes :
  - Le diagramme des cas d'utilisation (*Use Case Diagram*, indicateur **uc** ou **ucd**)
  - Le diagramme de séquence (*Sequence Diagram*, indicateur **seq**)
  - Le diagramme d'états (*State Machine Diagram*, indicateur **stm**)
  - Le diagramme d'activités (*Activity Diagram*, indicateur **act**)
- Un point de vue structurel, auxquels sont associés quatre diagrammes :
  - Le diagramme de définition de blocs (*Block Definition Diagram*, indicateur **bdd**)
  - Le diagramme de bloc interne (*Internal Block Diagram*, indicateur **ibd**)

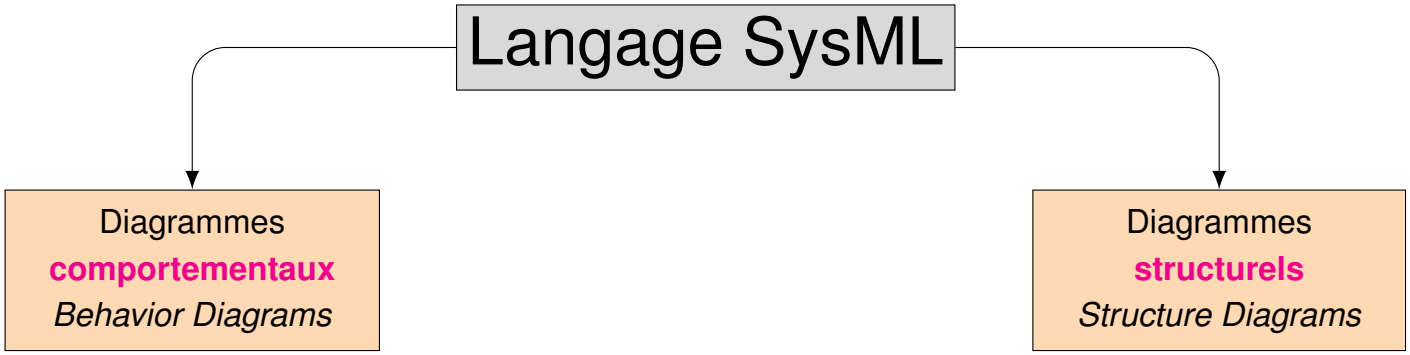
- Un point de vue comportemental, auxquels sont associés quatre diagrammes :
  - Le diagramme des cas d'utilisation (*Use Case Diagram*, indicateur **uc** ou **ucd**)
  - Le diagramme de séquence (*Sequence Diagram*, indicateur **seq**)
  - Le diagramme d'états (*State Machine Diagram*, indicateur **stm**)
  - Le diagramme d'activités (*Activity Diagram*, indicateur **act**)
- Un point de vue structurel, auxquels sont associés quatre diagrammes :
  - Le diagramme de définition de blocs (*Block Definition Diagram*, indicateur **bdd**)
  - Le diagramme de bloc interne (*Internal Block Diagram*, indicateur **ibd**)
  - Le diagramme paramétrique (*Parametric Diagram*, indicateur **par**)

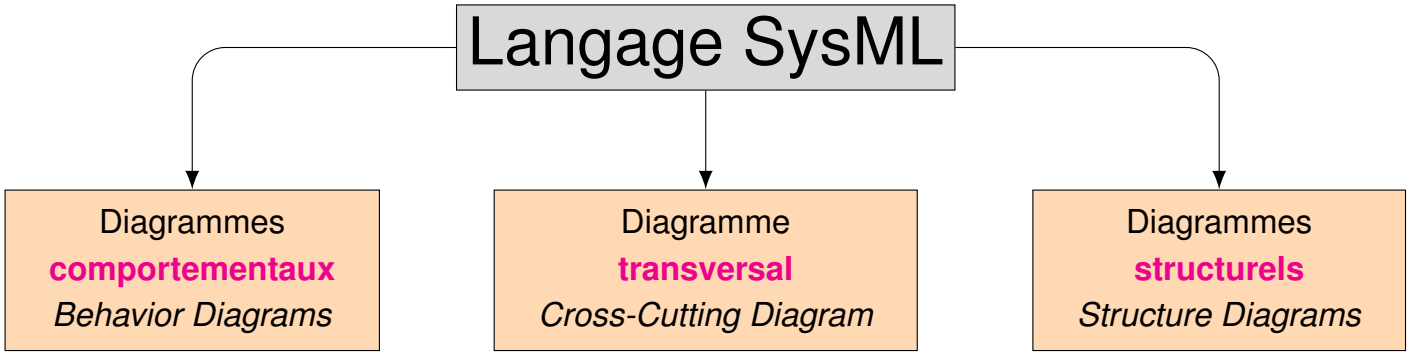
- Un point de vue comportemental, auxquels sont associés quatre diagrammes :
  - Le diagramme des cas d'utilisation (*Use Case Diagram*, indicateur **uc** ou **ucd**)
  - Le diagramme de séquence (*Sequence Diagram*, indicateur **seq**)
  - Le diagramme d'états (*State Machine Diagram*, indicateur **stm**)
  - Le diagramme d'activités (*Activity Diagram*, indicateur **act**)
- Un point de vue structurel, auxquels sont associés quatre diagrammes :
  - Le diagramme de définition de blocs (*Block Definition Diagram*, indicateur **bdd**)
  - Le diagramme de bloc interne (*Internal Block Diagram*, indicateur **ibd**)
  - Le diagramme paramétrique (*Parametric Diagram*, indicateur **par**)
  - Le diagramme de paquetages (*Package Diagram*, indicateur **pkg**)

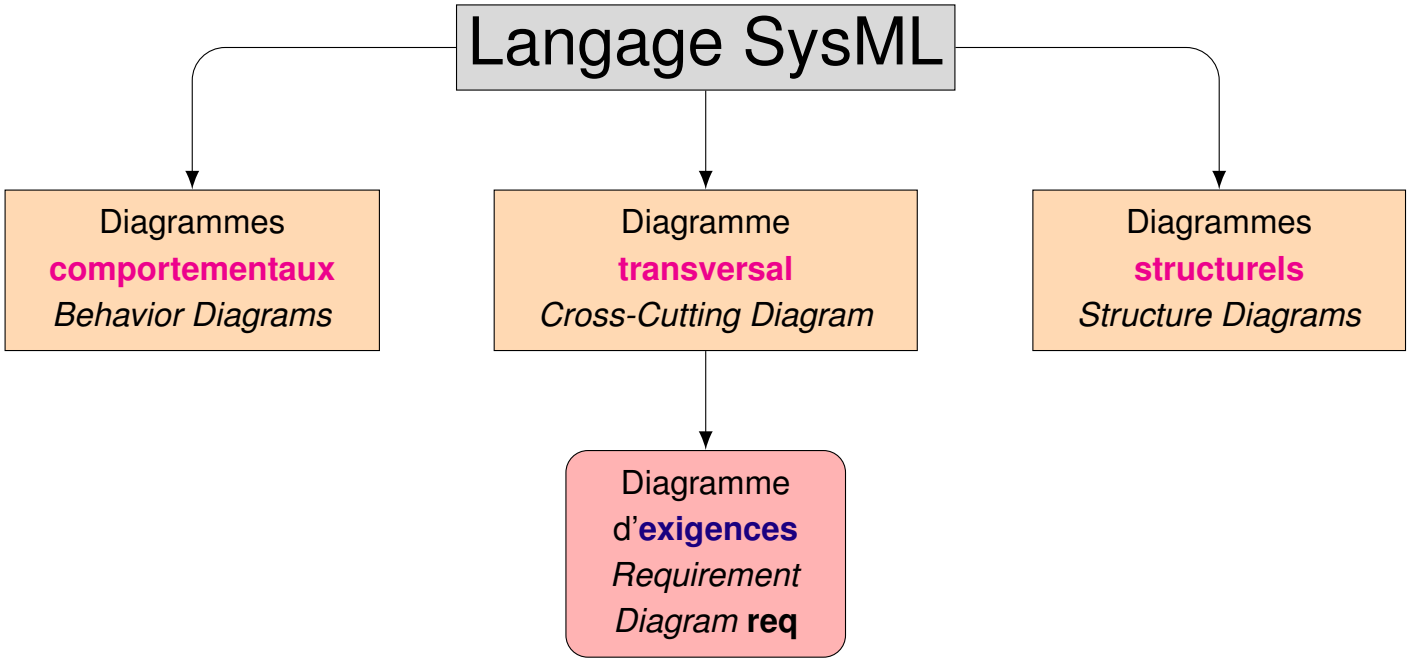
- Un point de vue comportemental, auxquels sont associés quatre diagrammes :
  - Le diagramme des cas d'utilisation (*Use Case Diagram*, indicateur **uc** ou **ucd**)
  - Le diagramme de séquence (*Sequence Diagram*, indicateur **seq**)
  - Le diagramme d'états (*State Machine Diagram*, indicateur **stm**)
  - Le diagramme d'activités (*Activity Diagram*, indicateur **act**)
- Un point de vue structurel, auxquels sont associés quatre diagrammes :
  - Le diagramme de définition de blocs (*Block Definition Diagram*, indicateur **bdd**)
  - Le diagramme de bloc interne (*Internal Block Diagram*, indicateur **ibd**)
  - Le diagramme paramétrique (*Parametric Diagram*, indicateur **par**)
  - Le diagramme de paquetages (*Package Diagram*, indicateur **pkg**)
- Un point de vue transversal, spécifique au langage SysML, a été rajouté : le diagramme d'exigences (*Requirement Diagram*, indicateur **req**)

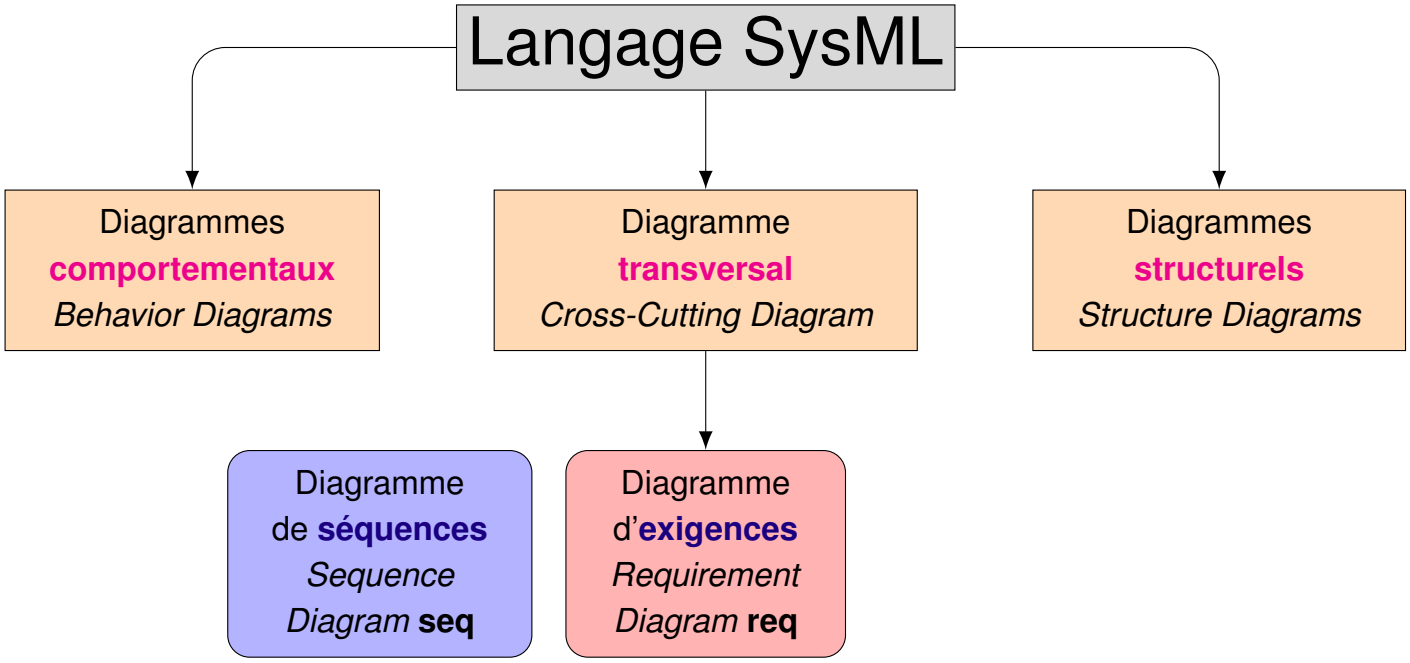
Langage SysML

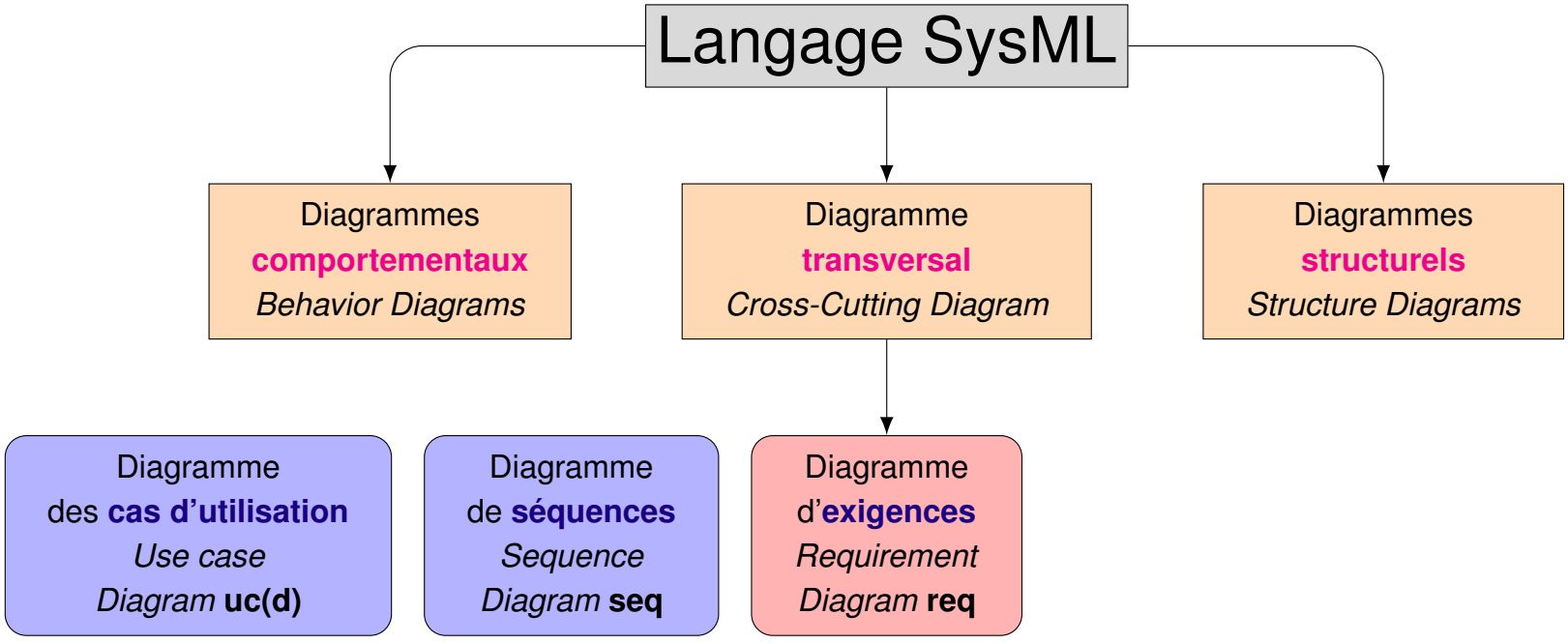

Langage SysML
Diagrammes
comportementaux
Behavior Diagrams

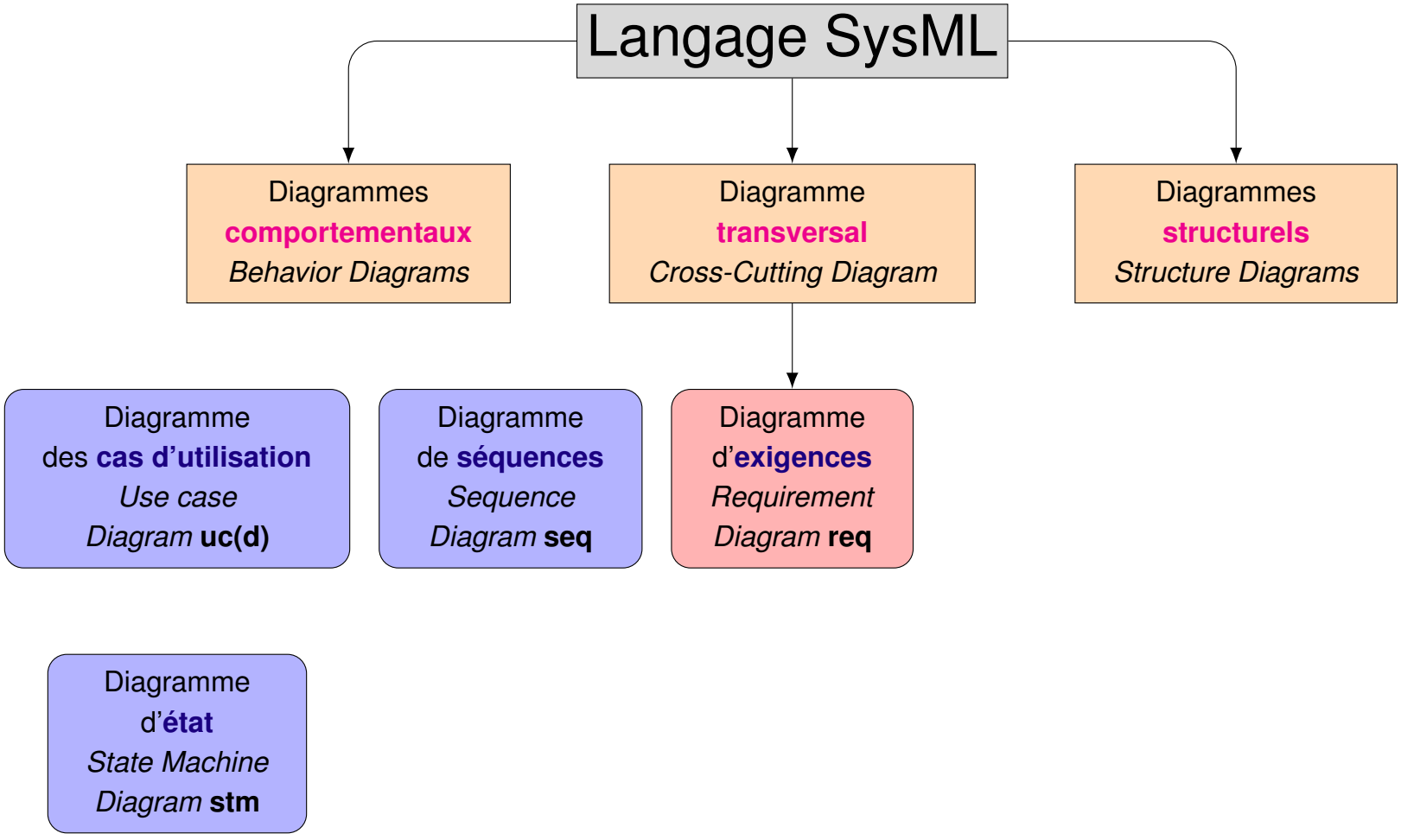

Langage SysML
Diagrammes
comportementaux
Behavior Diagrams
Diagrammes
structurels
Structure Diagrams

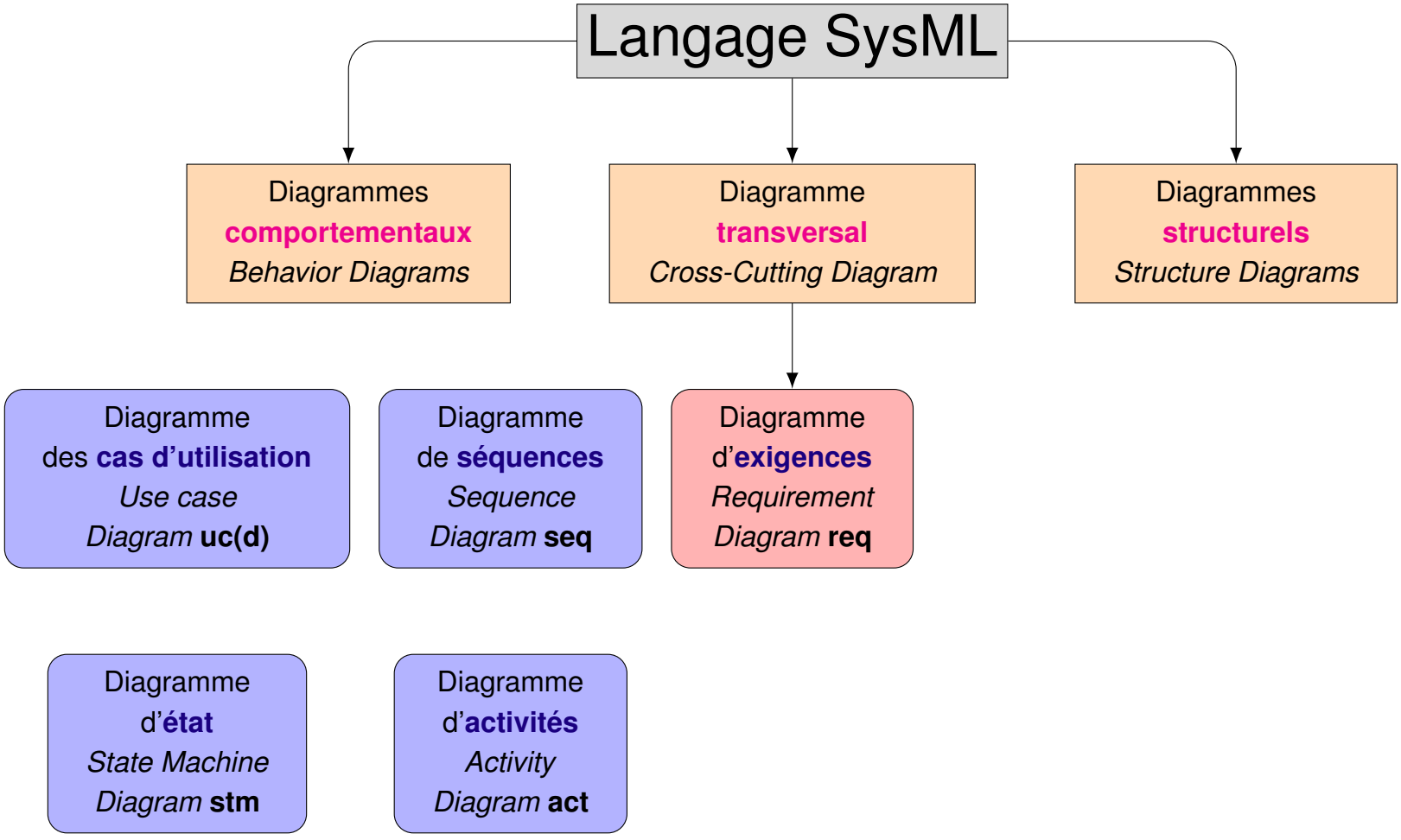

Langage SysML
Diagrammes
comportementaux
Behavior Diagrams
Diagramme
transversal
Cross-Cutting Diagram
Diagrammes
structurels
Structure Diagrams

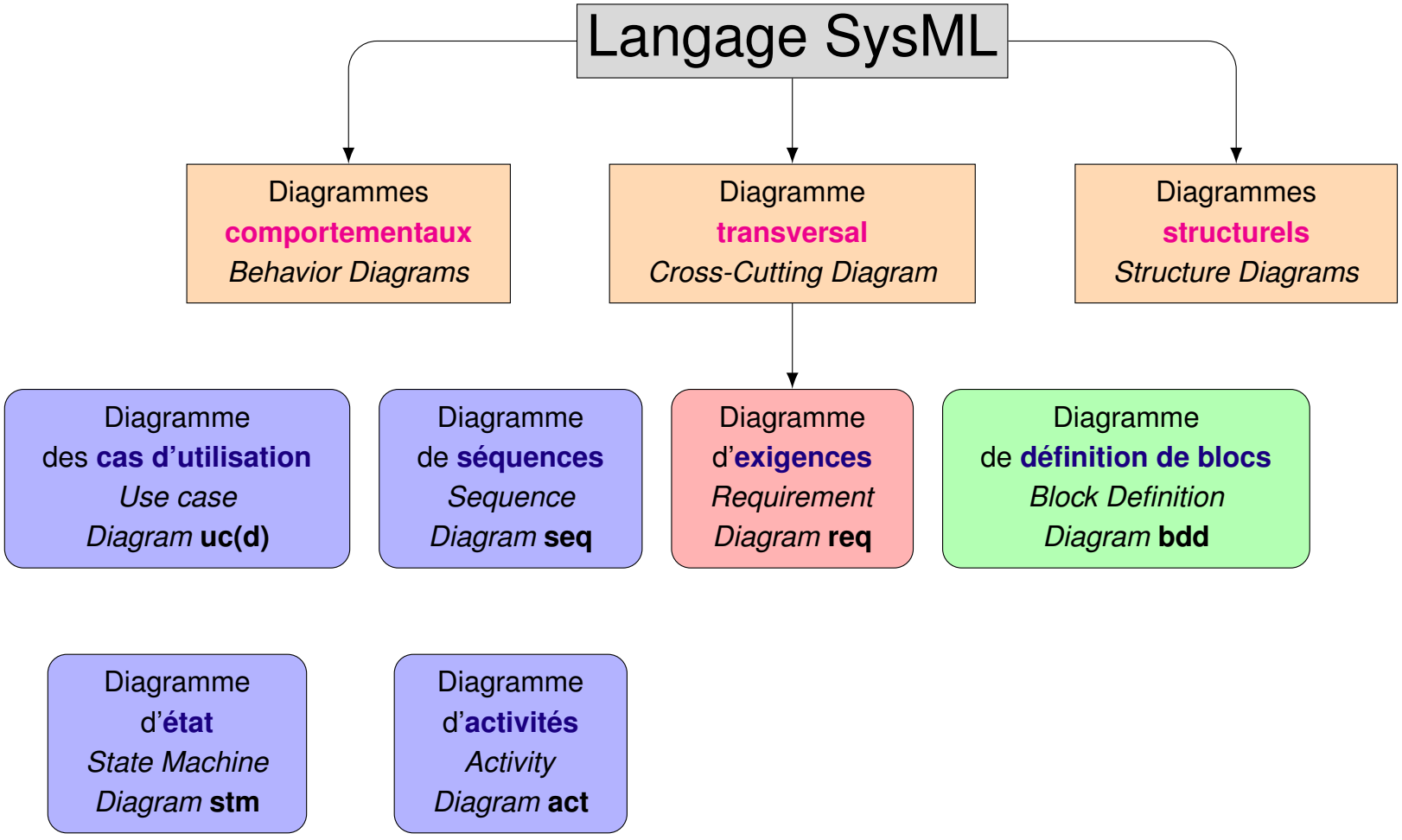

Langage SysML
Diagrammes
comportementaux
Behavior Diagrams
Diagramme
transversal
Cross-Cutting Diagram
Diagrammes
structurels
Structure Diagrams
Diagramme
d'exigences
Requirement
Diagram req

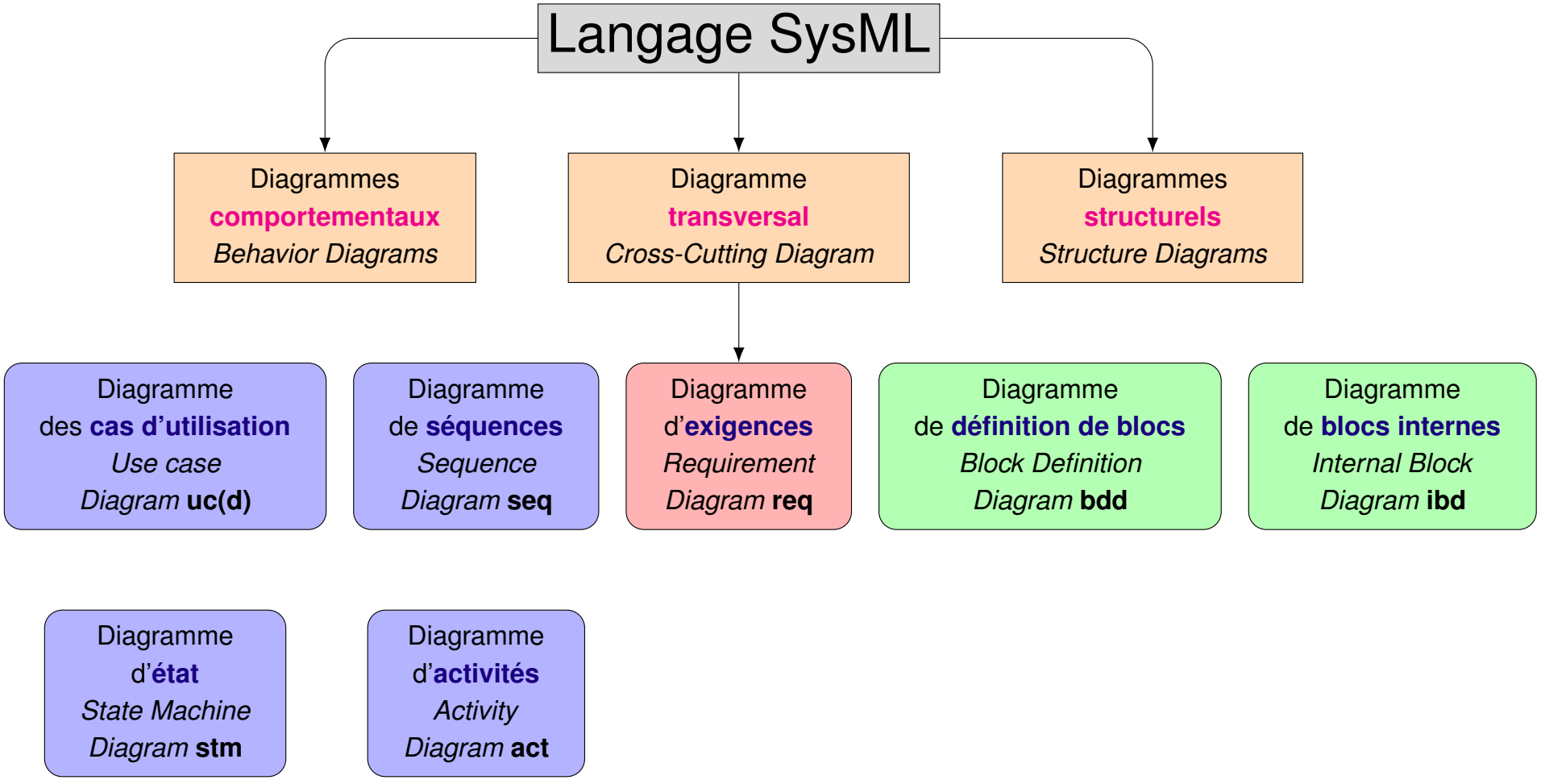

Langage SysML
Diagrammes
comportementaux
Behavior Diagrams
Diagramme
transversal
Cross-Cutting Diagram
Diagrammes
structurels
Structure Diagrams
Diagramme
de séquences
Sequence
Diagram seq
Diagramme
d'exigences
Requirement
Diagram req

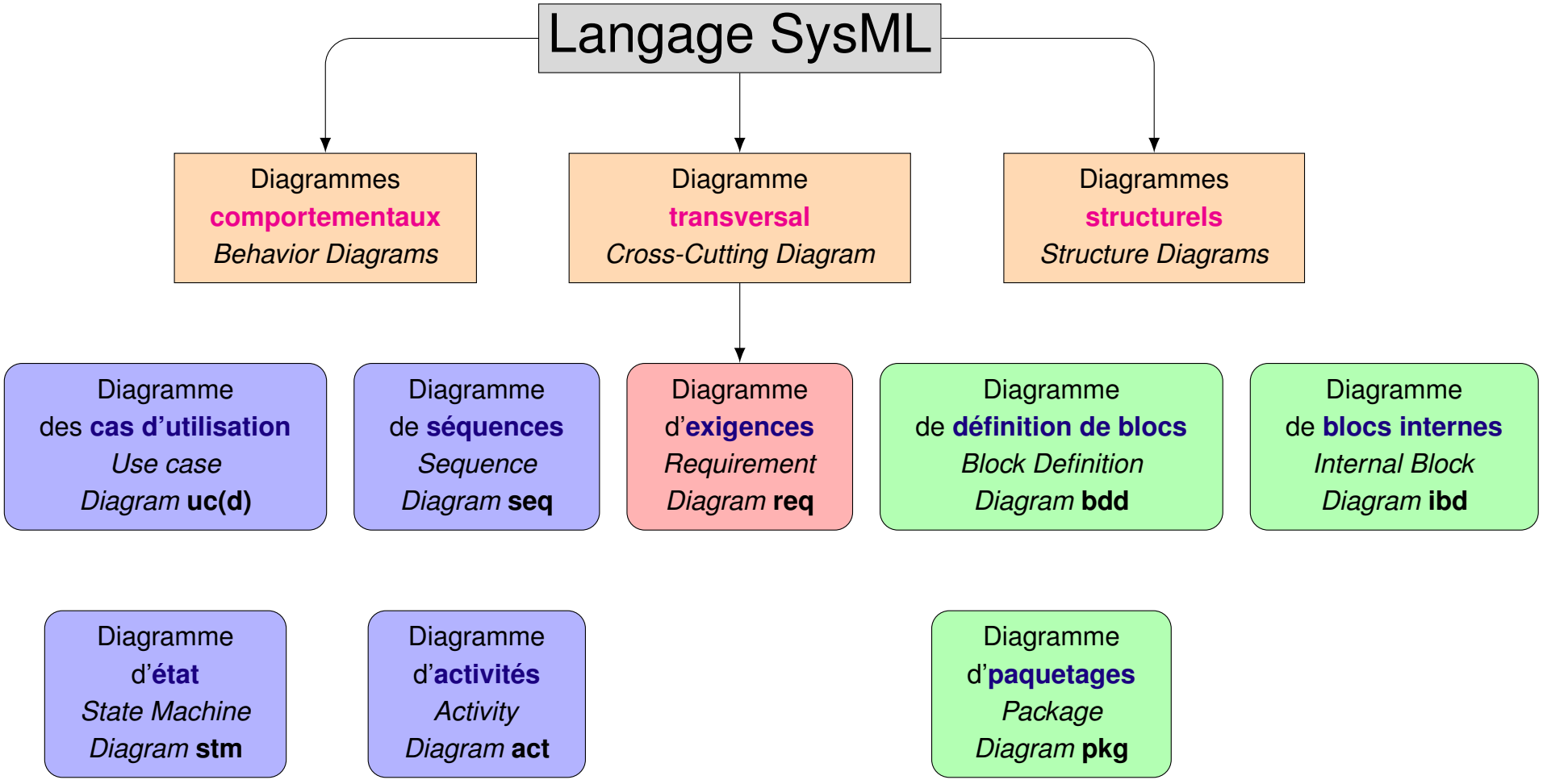

Langage SysML
Diagrammes
comportementaux
Behavior Diagrams
Diagramme
transversal
Cross-Cutting Diagram
Diagrammes
structurels
Structure Diagrams
Diagramme
des cas d'utilisation
Use case
Diagram uc(d)
Diagramme
de séquences
Sequence
Diagram seq
Diagramme
d'exigences
Requirement
Diagram req

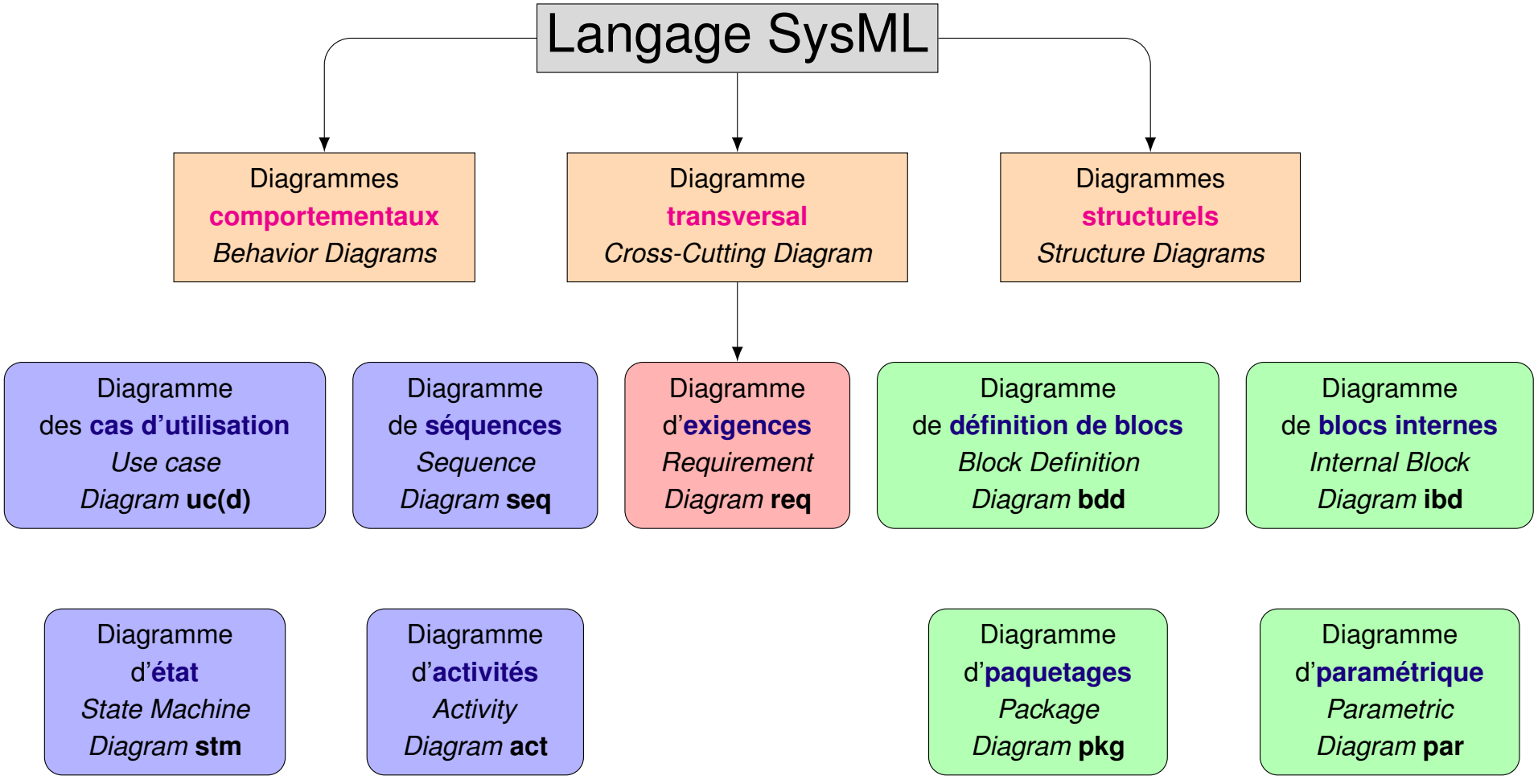

Langage SysML
Diagrammes
comportementaux
Behavior Diagrams
Diagramme
transversal
Cross-Cutting Diagram
Diagrammes
structurels
Structure Diagrams
Diagramme
des cas d'utilisation
Use case
Diagram uc(d)
Diagramme
de séquences
Sequence
Diagram seq
Diagramme
d'exigences
Requirement
Diagram req
Diagramme
d'état
State Machine
Diagram stm

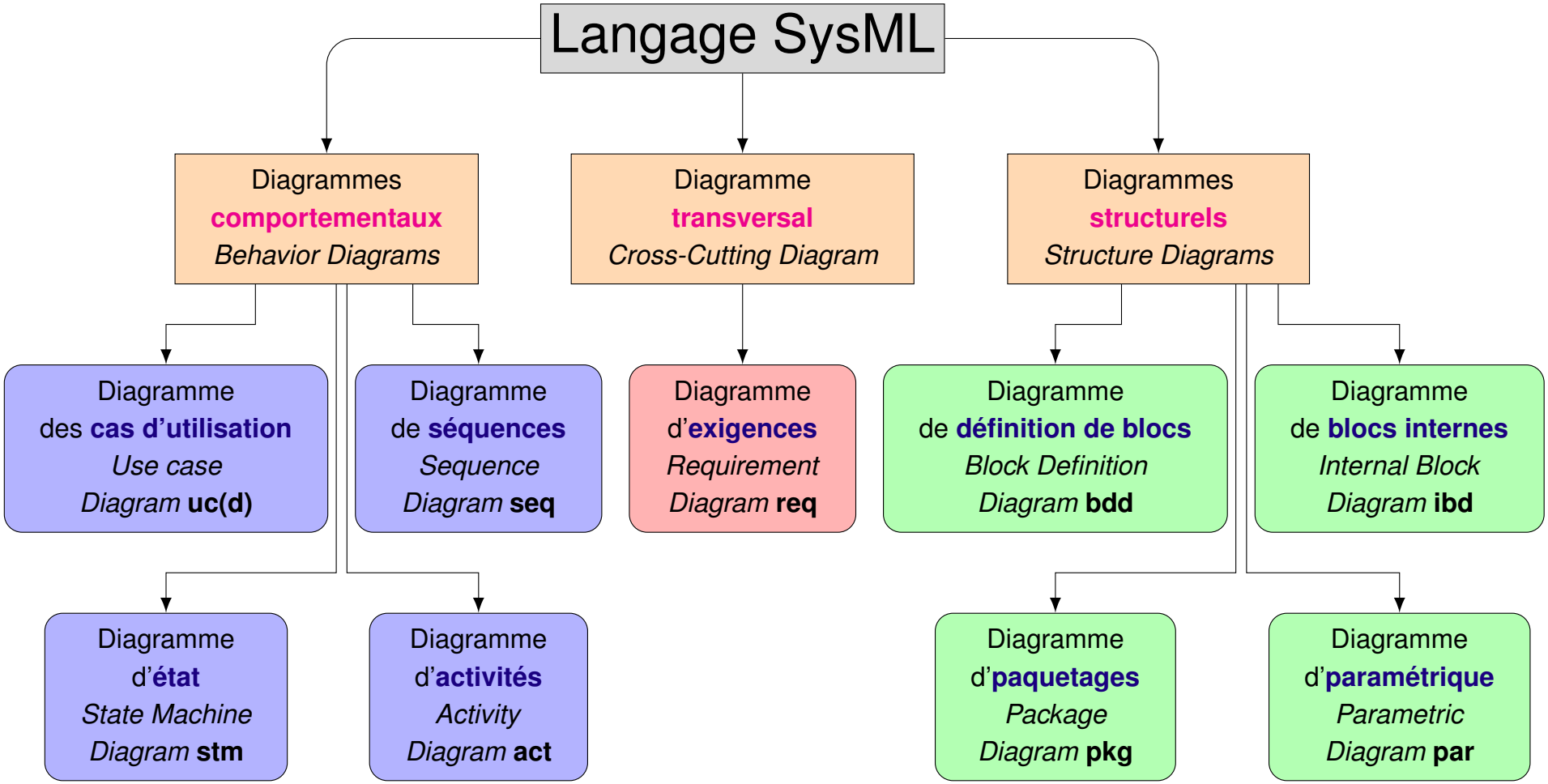

Langage SysML
Diagrammes
comportementaux
Behavior Diagrams
Diagramme
transversal
Cross-Cutting Diagram
Diagrammes
structurels
Structure Diagrams
Diagramme
des cas d'utilisation
Use case
Diagram uc(d)
Diagramme
de séquences
Sequence
Diagram seq
Diagramme
d'exigences
Requirement
Diagram req
Diagramme
d'état
State Machine
Diagram stm
Diagramme
d'activités
Activity
Diagram act

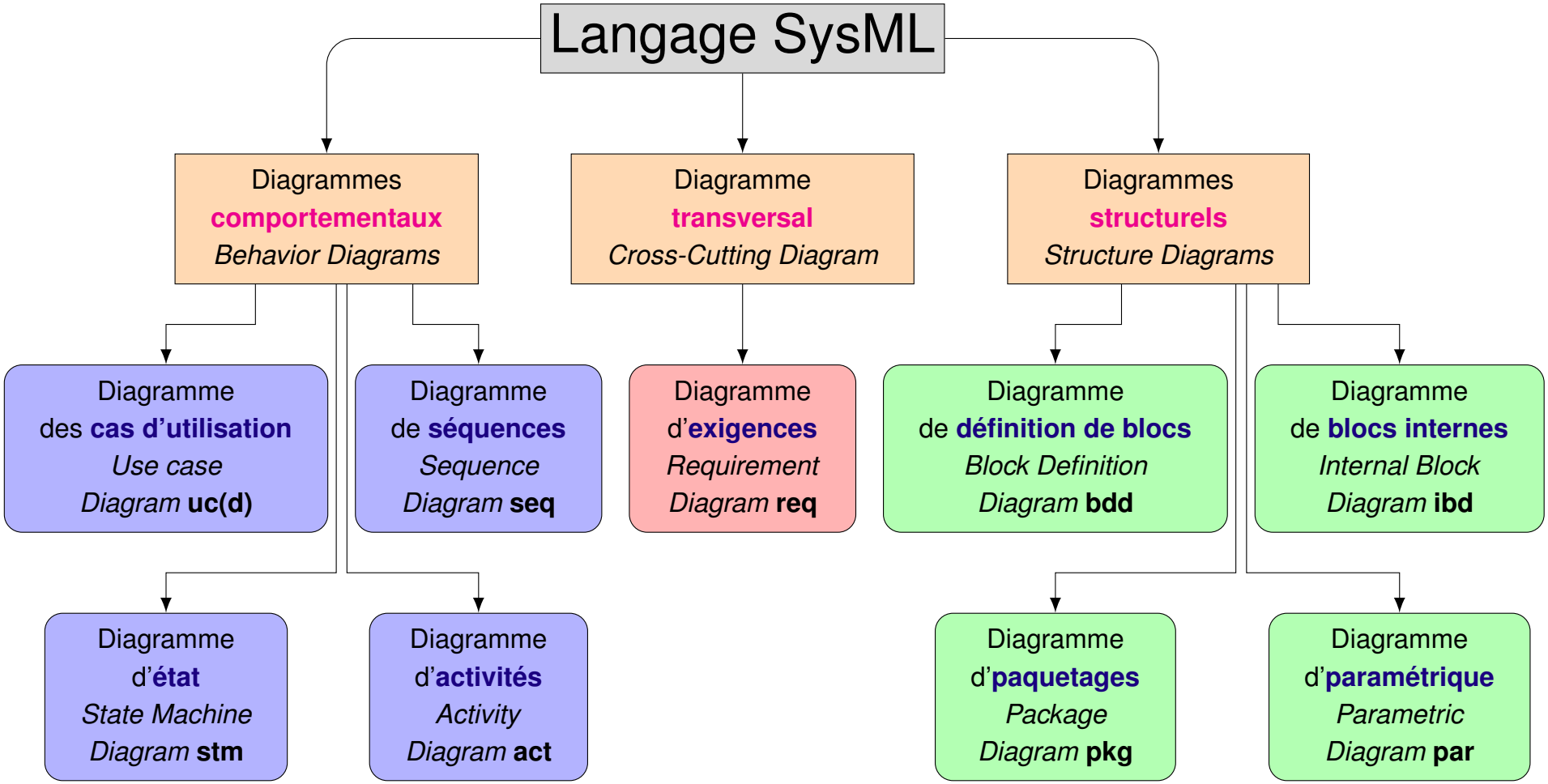

Langage SysML
Diagrammes
comportementaux
Behavior Diagrams
Diagramme
transversal
Cross-Cutting Diagram
Diagrammes
structurels
Structure Diagrams
Diagramme
des cas d'utilisation
Use case
Diagram uc(d)
Diagramme
de séquences
Sequence
Diagram seq
Diagramme
d'exigences
Requirement
Diagram req
Diagramme
de définition de blocs
Block Definition
Diagram bdd
Diagramme
d'état
State Machine
Diagram stm
Diagramme
d'activités
Activity
Diagram act


Langage SysML
Diagrammes
comportementaux
Behavior Diagrams
Diagramme
transversal
Cross-Cutting Diagram
Diagrammes
structurels
Structure Diagrams
Diagramme
des cas d'utilisation
Use case
Diagram uc(d)
Diagramme
de séquences
Sequence
Diagram seq
Diagramme
d'exigences
Requirement
Diagram req
Diagramme
de définition de blocs
Block Definition
Diagram bdd
Diagramme
de blocs internes
Internal Block
Diagram ibd
Diagramme
d'état
State Machine
Diagram stm
Diagramme
d'activités
Activity
Diagram act


Langage SysML
Diagrammes
comportementaux
Behavior Diagrams
Diagramme
transversal
Cross-Cutting Diagram
Diagrammes
structurels
Structure Diagrams
Diagramme
des cas d'utilisation
Use case
Diagram uc(d)
Diagramme
de séquences
Sequence
Diagram seq
Diagramme
d'exigences
Requirement
Diagram req
Diagramme
de définition de blocs
Block Definition
Diagram bdd
Diagramme
de blocs internes
Internal Block
Diagram ibd
Diagramme
d'état
State Machine
Diagram stm
Diagramme
d'activités
Activity
Diagram act
Diagramme
d'paquetages
Package
Diagram pkg


Langage SysML
Diagrammes
comportementaux
Behavior Diagrams
Diagramme
transversal
Cross-Cutting Diagram
Diagrammes
structurels
Structure Diagrams
Diagramme
des cas d'utilisation
Use case
Diagram uc(d)
Diagramme
de séquences
Sequence
Diagram seq
Diagramme
d'exigences
Requirement
Diagram req
Diagramme
de définition de blocs
Block Definition
Diagram bdd
Diagramme
de blocs internes
Internal Block
Diagram ibd
Diagramme
d'état
State Machine
Diagram stm
Diagramme
d'activités
Activity
Diagram act
Diagramme
d'paquetages
Package
Diagram pkg
Diagramme
d'paramétrique
Parametric
Diagram par


Langage SysML
Diagrammes comportementaux Behavior Diagrams
Diagramme transversal Cross-Cutting Diagram
Diagrammes structurels Structure Diagrams
Diagramme des cas d'utilisation Use case Diagram uc(d)
Diagramme de séquences Sequence Diagram seq
Diagramme d'exigences Requirement Diagram req
Diagramme de définition de blocs Block Definition Diagram bdd
Diagramme de blocs internes Internal Block Diagram ibd
Diagramme d'état State Machine Diagram stm
Diagramme d'activités Activity Diagram act
Diagramme d'paquetages Package Diagram pkg
Diagramme d'paramétrique Parametric Diagram par



Ce n'est pas une méthode, il n'y a pas obligatoirement d'ordre dans l'établissement des diagrammes. Cependant, il y a tout de même une manière *naturelle* de procéder.

diagramme d'exigence (**req**)

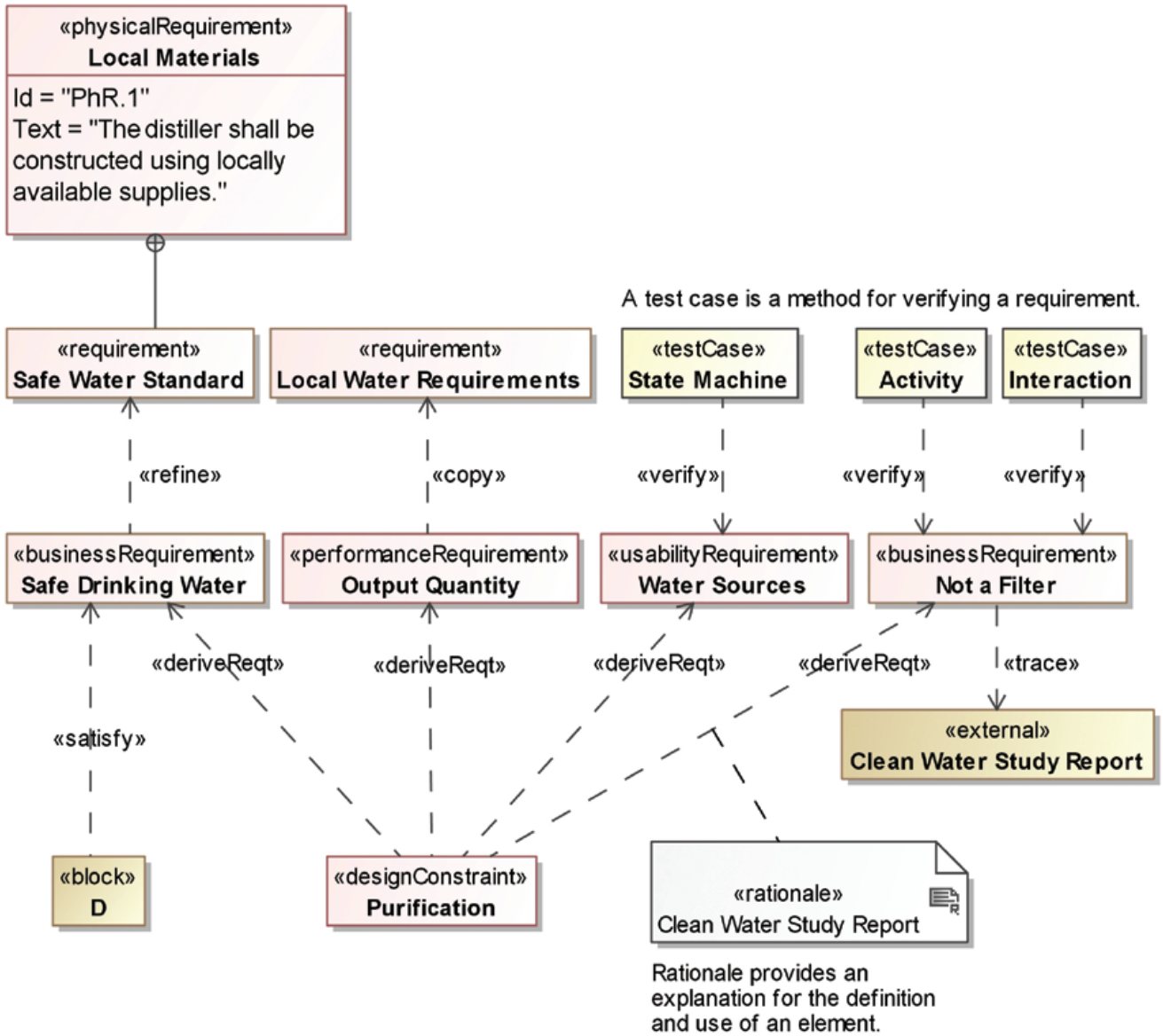


diagramme d'exigence (**req**)

Requirement:
A Requirement specifies a capability or a condition that must (or should) be satisfied. Requirements are used to establish a contract between the customer (or other stakeholders) and those responsible for designing and implementing the system. A requirement can also appear on other diagrams to show its relationship to other modeling elements.

Extended Requirement:
An Extended Requirement adds some properties to the requirement element. These properties are important for requirement management. Specific projects should add their own properties.

Functional Requirement:
A Functional Requirement is a requirement that specifies a behavior that a system or part of a system must perform.

Interface Requirement:
An Interface Requirement is a requirement that specifies the ports for connecting systems and parts of a system. Optionally, it may include the items that flow across the connector and/or the Interface constraints.

Performance Requirement:
A Performance Requirement refers to a requirement that quantitatively measures the extent to which a system or a system part satisfy a required capability or condition.

Physical Requirement:
A Physical Requirement specifies the physical characteristics and/or physical constraints of a system, or a system part

Design Constraint:
A Design Constraint is a requirement that specifies a constraint on the implementation of a system or on part of it.

Business Requirement:
A Business Requirement is a requirement that specifies characteristics of the business process that must be satisfied by the system.

Usability Requirement:
A Usability Requirement specifies the fitness for use of a system for its users and other actors.

Containment:
This relationship is used to decompose a requirement into its constituent requirements.

Allocation:
an allocation relationship to allocate one model element to another.

diagramme d'exigence (**req**)

Requirements Dependencies
Trace:
A 'Trace' relationship is a dependency that provides a general purpose relationship between a requirement and any other model elements.
Satisfy:
A 'Satisfy' relationship is a dependency between a requirement and a model element that fulfills that requirement. As with other dependencies, the arrow direction points from the satisfying (client) model element to the (supplier) requirement that is satisfied.
Copy:
A 'Copy' relationship is a dependency between a supplier requirement (master) and a client requirement (slave), specifying that the client requirement text is a read-only copy of the supplier requirement text.

Verify:
A 'Verify' relationship is a dependency between a requirement and a test case or a model element that can determine whether the system fulfills the requirement. Other dependencies, the arrow direction points from the (client) test case to the (supplier) requirement.
Derive:
A 'Derive' relationship is a dependency between two requirements (a derived requirement and a source requirement), where the derived requirement is generated or inferred from the source requirement.
Refine:
A 'Refine' relationship is a dependency intended to describe how a model element or a set of elements are used to further refine a requirement. Alternatively, it can be used to show how a text-based requirement refines a model element.

diagramme des cas d'utilisation (**uc**)

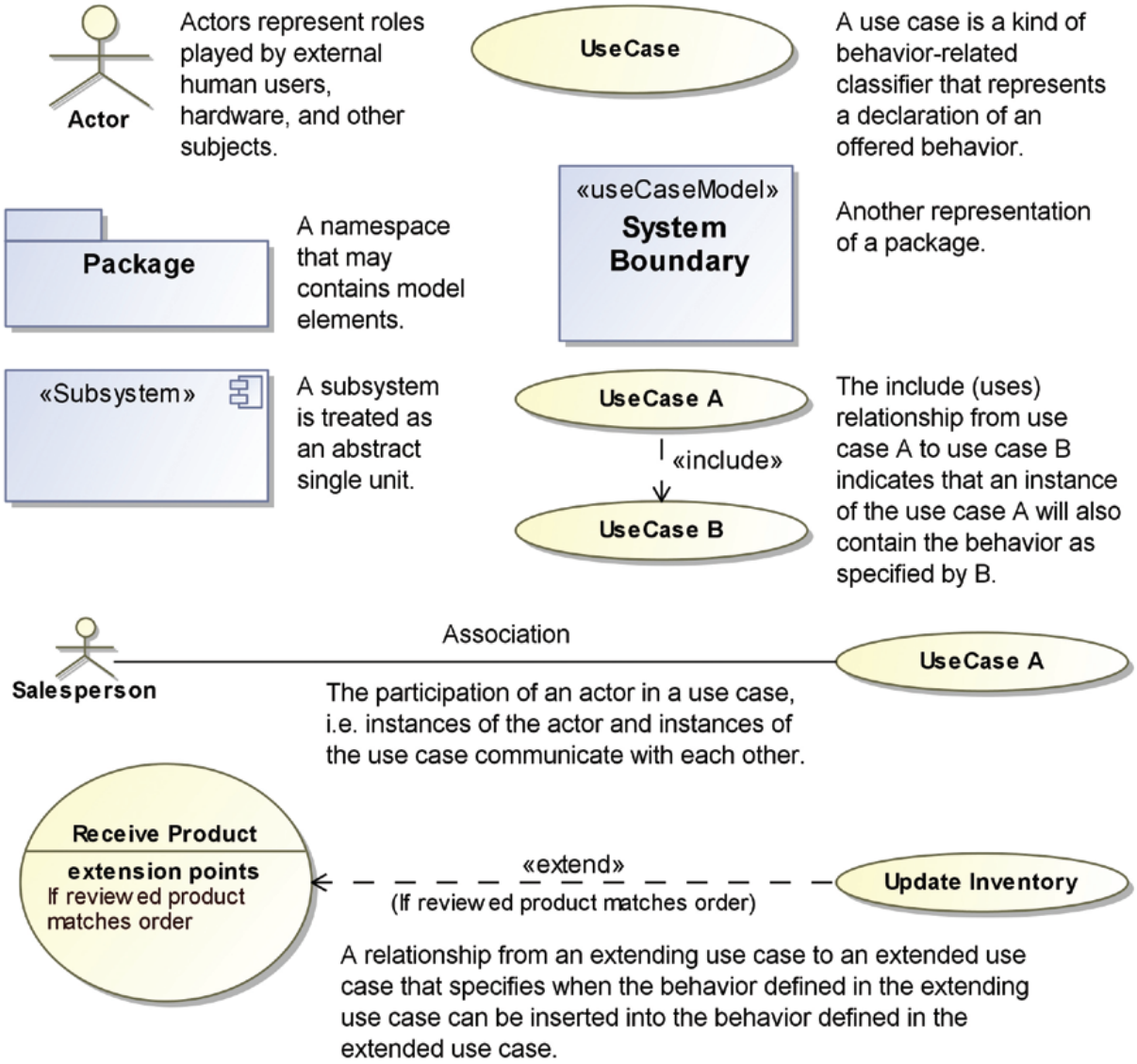


diagramme de séquences (**sd**)

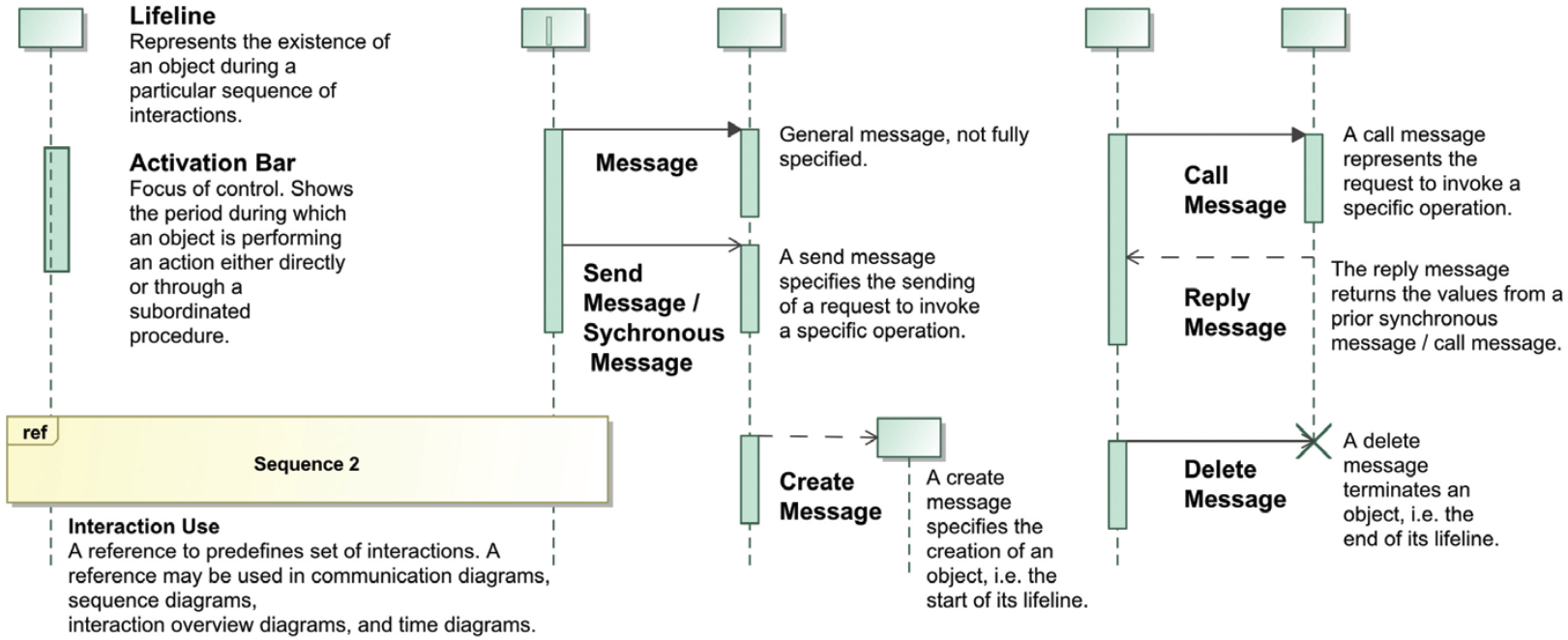


diagramme de séquences (**sd**)

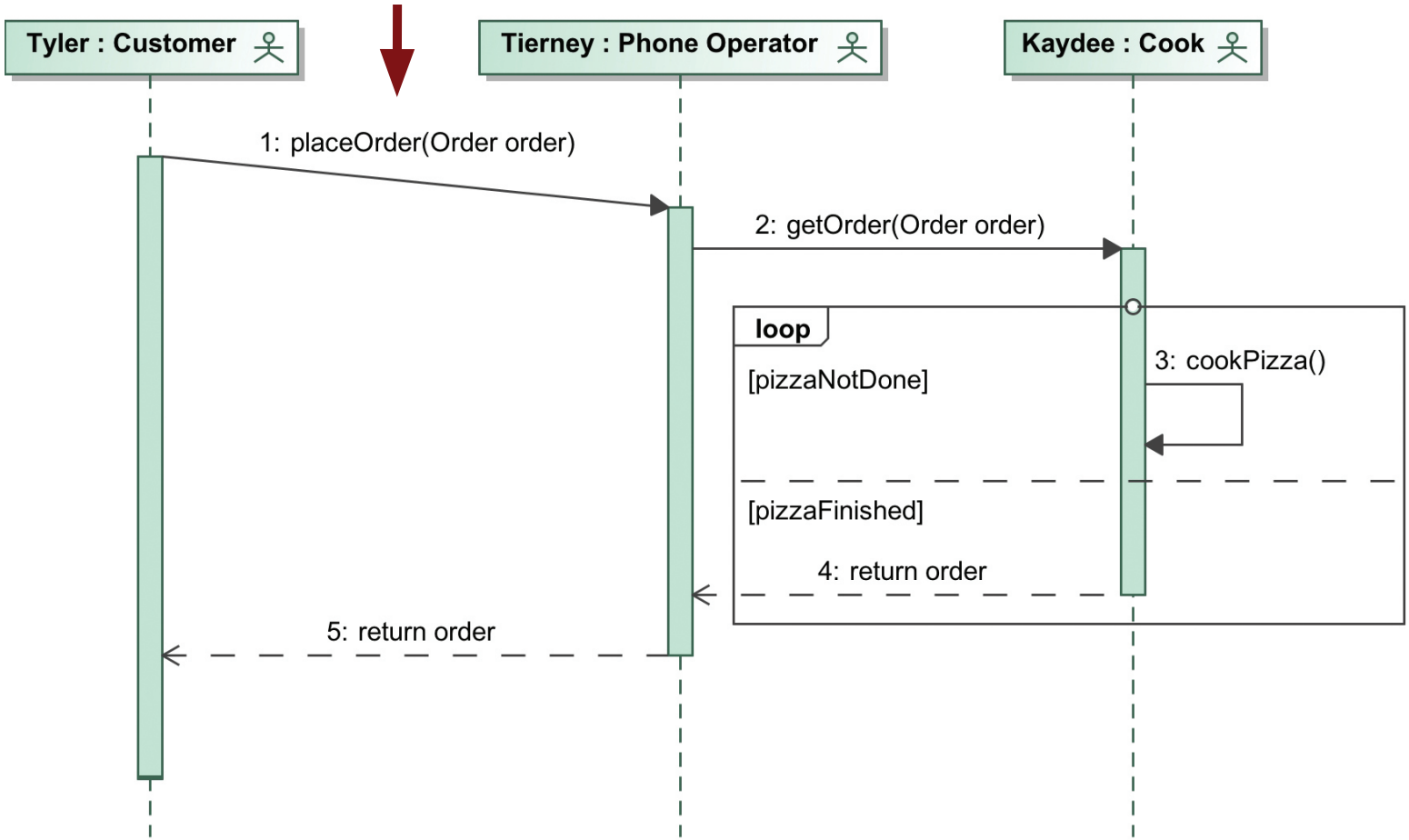
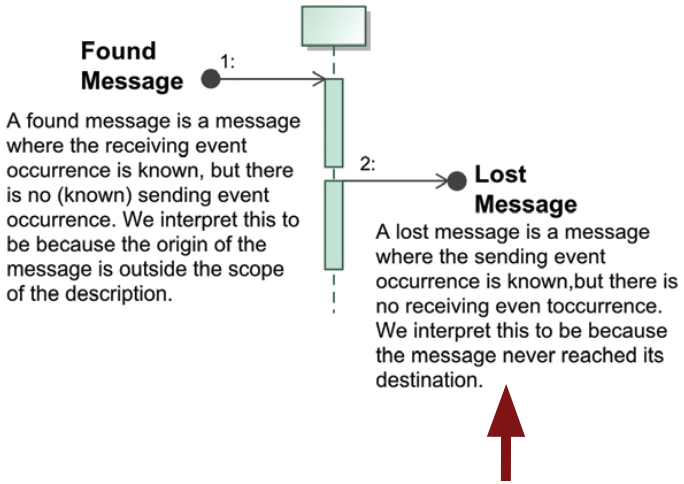
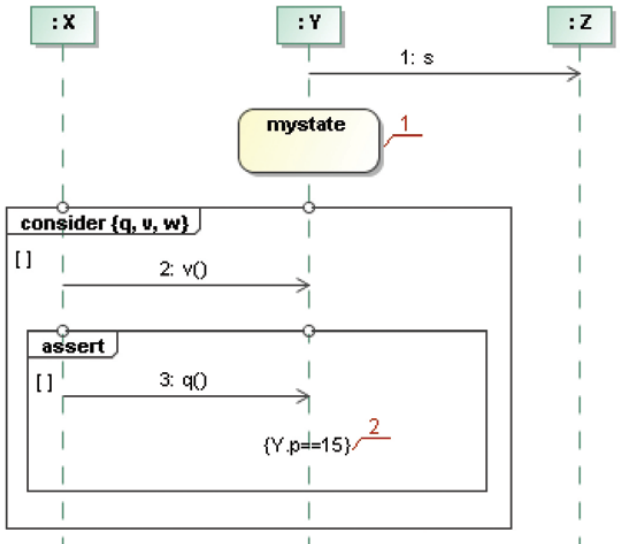


diagramme de séquences (**sd**)





State Invariant

A StateInvariant is a runtime constraint on the participants of the interaction. It may be used to specify a variety of different kinds of constraints, such as values of attributes or variables, internal or external states, and soon.

diagramme de séquences (**sd**)

loop

[] **Loop**
The loop combined fragment represents that the loop operand will be repeated a number of times. If the loop contains a separate interaction constraint with a specification, the loop will only continue if that specification evaluates to true during execution regardless of the minimum number of iterations specified in the loop.

break

[] **Break**
The interaction operator brk designates that the combined fragment represents a breaking scenario in the sense that the operand is a scenario that is performed instead of the remainder of the enclosing interaction fragment. A break operator with a guard is chosen when the guard is true and the rest of the enclosing interaction fragment is ignored.

opt

[] **Option**
The option combined fragment opt represents a choice of behavior where either the (sole)operand happens or nothing happens. An option combined fragment is used to model "if-then" construct.

neg

[] **Negative**
The interaction operator neg designates that the combined fragment represents traces that are defined to be invalid. The set of traces that defined a combined fragment with interaction operator negative is equal to the set of traces given by its (sole) operand, only that this set is a set of invalid rather than valid traces.

alt

[] **Alternatives**
The alternative combined fragment alt represents a choice of behavior. Alternative combined fragment has several operands. At most one of the operands has to be chosen.Using alternative combined fragment you can model if-then-else statement.

[else]

par

[] **Parallel**
The interaction operator par designates that the combined fragment represents a parallel merge between the behaviors of the operands. A parallel merge defines a set of traces that describes all the ways that occurrence specifications of the operands may be interleaved without obstructing the order of the occurrence specifications within the operand.

seq

[] **Weak Sequencing**
The interaction operator seq designates that the combined fragment represents a weak sequencing between the behaviors of the operands. It is the same as parallel execution,except that event on the same lifeline from different sub fragments are ordered in the same order as the sub fragments within the enclosing weak sequencing fragment.

consider

[] **Consider**
The interaction operator con designates which messages should be considered within this combined fragment. This is equivalent to defining every other message to be ignored.

ignore

[] **Ignore**
The interaction operator ign designates that there are some message types that are not shown within this combined fragment. These message types can be considered insignificant and are implicitly ignored if they appear in a corresponding execution.

critical

[] **Critical Region**
The interaction operator crt designates that the combined fragment represents a critical region. A critical region means that the traces of the region cannot be interleaved by other occurrence specifications (on those Lifelines covered by the region).

strict

[] **Strict Sequencing**
The interaction operator str designates that the combined fragment represents a strict sequencing between the behaviors of the operands. The semantics of strict sequencing defines a strict ordering of the operands on the first level within the combined fragment with interaction operator strict.

assert

[] **Assertion**
The interaction operator asr designates that the combined fragment represents an assertion. The sequences of the operand of the assertion are the only valid continuations.All other continuations result in an invalid trace.

diagramme de définition de blocs (**bdd**)

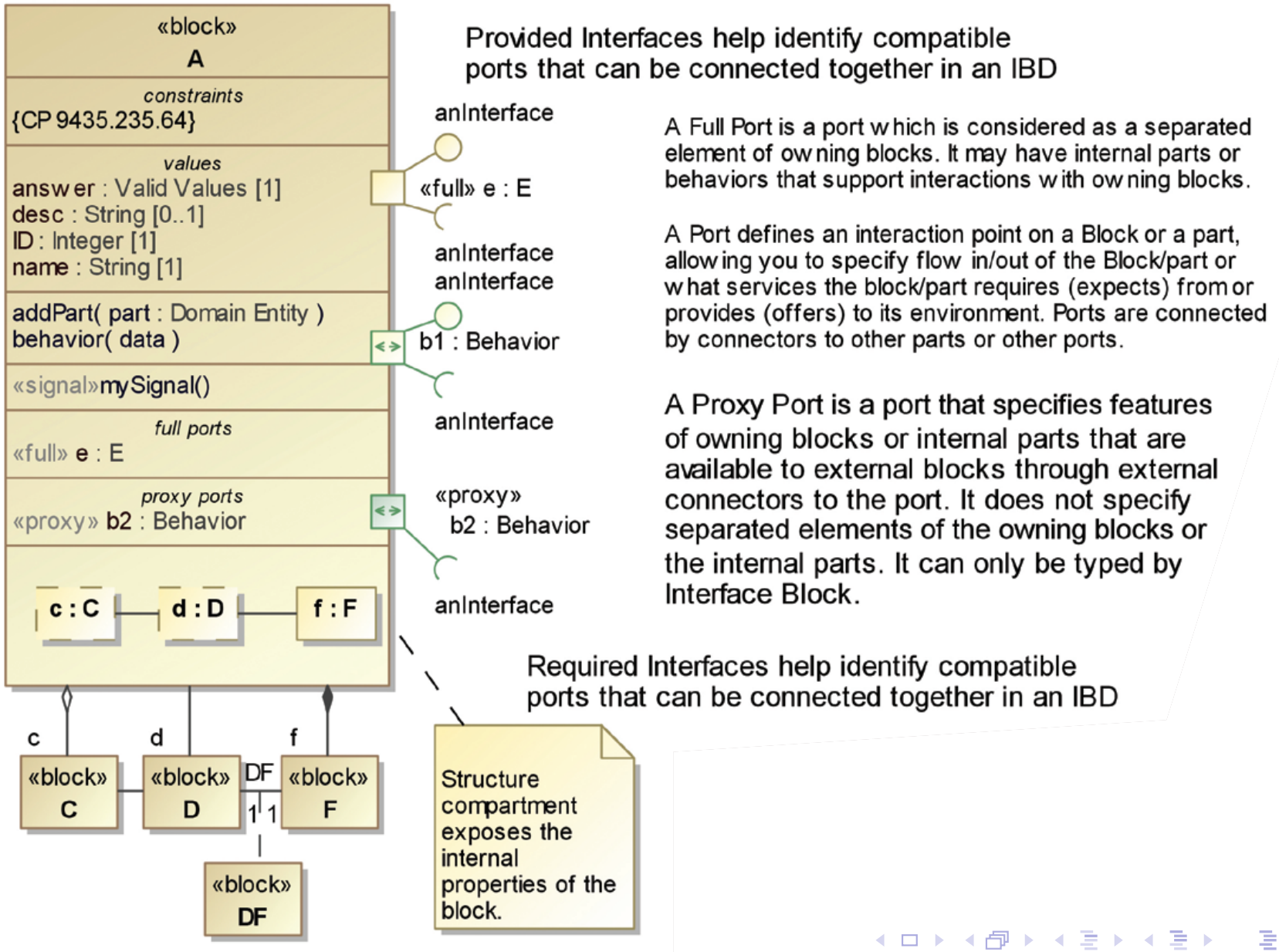
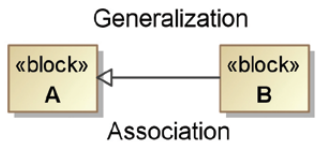
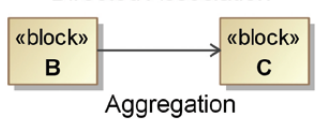


diagramme de définition de blocs (**bdd**)

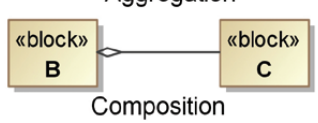


A relationship between a more general and a more specific element.

A connection among classes, which also means a connection among objects of those classes.

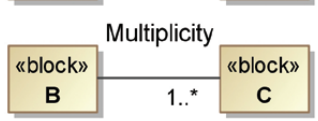


Directed associations indicate unidirectional visibility, i.e. B has access to C, not vice versa.

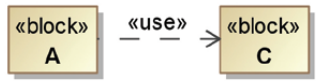


An aggregation is an association that represents a whole-part relationship.

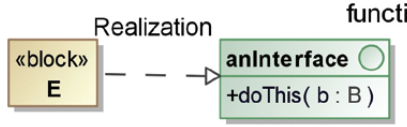
A composition is a form of aggregation with coincident lifetime of part with the whole.



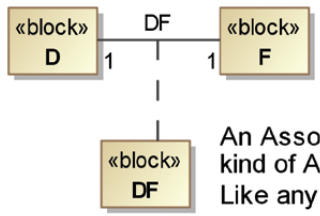
Defines the limits on the number of objects of each type that may participate in the association.



A Usage is a dependency in which one element (the client) requires the presence of another element (the supplier) for its correct functioning or implementation.

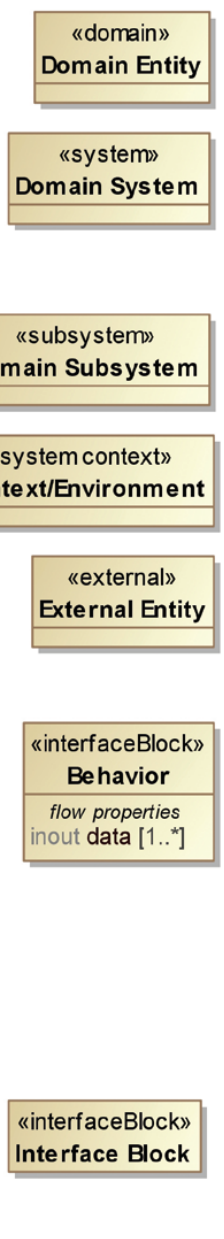


A realization signifies that the client set of elements are an implementation of the supplier set, which serves as the specification.



An Association Block is an Association Class (a kind of Association) stereotyped by «Block». Like any other Block, an Association Block can own properties and connectors.

diagramme de définition de blocs (**bdd**)



A Domain block represents an entity, a concept, a location, or a person from the real-world domain. A domain block is part of the system knowledge.

A System is an artificial artifact consisting of blocks that pursue a common goal which cannot be achieved by the system's individual elements. A block can be a software, hardware, a person, or an arbitrary unit.

A Subsystem is a typically large, encapsulated block within a larger system.

A System context element is a virtual container that includes the entire system and its actors

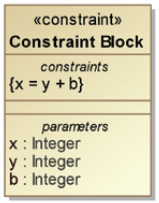
An External block is a block that represents an actor. It facilitates a more detailed modeling of actors like ports or internal structures.

An Interface Block is a special kind of block which has no behaviors or internal parts. It is used to type proxy ports.

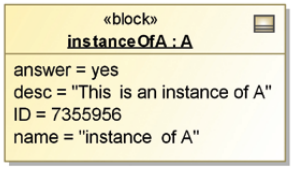
A FlowProperty signifies a single flow element that can flow to/from a block. Flow properties are defined directly on blocks or flow specifications that are those specifications which type the flow ports. Flow properties enable item flows across connectors connecting parts of the corresponding block types. A flow property's values are either received from or transmitted to an external block.

An Interface specifies operations or signals. If an Interface is provided to a port, the external parts may call operations or send signals to the Block owning the port via that port. If an Interface is required for a port, the Block owning

diagramme de définition de blocs (**bdd**)

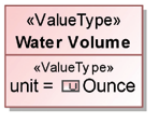


Constraint Blocks provide a mechanism to integrate engineering analysis, such as performance and reliability models, with other SysML models. Constraint Blocks can be used to specify a network of constraints representing mathematical expressions, which constrain the physical properties of a system. Constraint Blocks are generally defined in Block Definition Diagrams, and then used in Parametric diagrams.

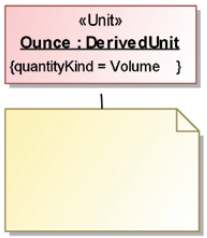


An Instance Specification is one incarnation of the Block specification in which properties are given values.

Value Types
In general, define quantity kinds first, followed by units and their quantity kinds. After that, define value types and their units (and quantity kinds).



A Value Type is defined as a stereotype of UML Data Type to establish a more neutral term for system values that may never be given a concrete data representation. A Value Type adds an ability to carry a unit of measure of a quantity kind associated with the value.



A Unit is a quantity in terms of which the magnitudes of other quantities that have the same quantity kind can be stated. A unit often relies on precise and reproducible ways of measuring the unit. For example, a unit of length such as meter may be specified as a multiple of a particular wavelength of light. A unit may also specify less stable or precise ways to express some value, such as a cost expressed in some currency, or a severity rating measured by a numerical scale.

Notes may display tagged values of the anchoring element.

A Quantity Kind (formerly 'Dimension' in SysML 1.0 and 1.1 specifications) is a kind of quantity that may be stated by means of defined units. For example, the quantity kind of length can be measured by units of meters, kilometers, or feet.

An Enumeration is a kind of Data Type whose instances may be any of the user-predefined enumeration literals. It is possible to extend the set of applicable enumeration literals to other packages or profiles.

A Data Type is a type whose instances are identified only by their values. A typical use of Data Types would be to represent the primitive types of the programming language used. For example, integer and string types are often treated as data types.

diagramme de définition de blocs (**bdd**)

Constraints and Constraint Properties.
Constraint Properties:
Properties which are typed by Constraint Blocks, or subtypes of Constraint Block, always having 'composite' aggregation kind.
Value Properties:
Properties which are typed by Value Types or subtypes of Value Type, always having 'composite' aggregation kind.
Part Properties:
Properties which are typed by Blocks or subtypes of Block, except Constraint Block, having 'composite' aggregation kind.
Reference Properties:
Properties which are typed by Blocks or subtypes of Block, except Constraint Block, having 'shared' and 'none' aggregation kind, respectively.
FlowProperty:
A FlowProperty signifies a single flow element that can flow to/from a block. Flow properties are defined directly on blocks or flow specifications that are those specifications which tie the flow ports.
Flow Properties:
Flow properties enable item flows across connectors connecting parts of the corresponding block types, either directly (in the case of the properties that are defined on the block) or via flowPorts. A flow property's values are either received from or transmitted to an external block.

Chaine fonctionnelle