The Generic SysML/KAOS Domain Metamodel

Steve Jeffrey Tueno Fotso and Marc Frappier

GRIL, Université de Sherbrooke, Sherbrooke, QC J1K 2R1, Canada

Régine Laleau

LACL, Université Paris-Est Créteil, 94010, CRÉTEIL, France

Amel Mammar

SAMOVAR-CNRS, Télécom SudParis, 91000, Evry, France

Abstract

This paper is related to the generalised/generic version of the SysML/KAOS domain metamodel and on translation rules between the new domain models and *B System* specifications.

Keywords: Requirements Engineering, Domain Modeling, *SysML/KAOS*, Ontologies, *B System*, *Event-B*

1. Event-B and B System

Event-B [1] is an industrial-strength formal method for *system modeling*. It is used to incrementally construct a system specification, using refinement, and to prove useful properties. *B System* is an *Event-B* syntactic variant proposed by *ClearSy*, an industrial partner in the *FORMOSE* project [2], and supported by *Atelier B* [3]. *Event-B* and *B System* have the same semantics defined by proof obligations [1].

Figure 1 is a metamodel of the *B System* language restricted to concepts that are relevant to us. A *B System* specification consists of components (instances of Component). Each component can be either a system or a refinement and it may define static or dynamic elements. A refinement is a component which refines another one in order to access the elements defined in it and to reuse them for new constructions. Constants, abstract and enumerated sets, and their properties, constitute the static part. The dynamic part includes the representation of the system state using variables constrained through invariants and initialised through initialisation actions. Properties and invariants can be categorised as instances of LogicFormula. Variables can be involved only in invariants. In our case, it is sufficient to consider that logic formulas are successions of operands in relation through operators. Thus, an instance of LogicFormula references its operators (instances of Operator) and its operands that may be instances of Variable, Constant, Set or SetItem.

2. The SysML/KAOS Domain Modeling Language

Domain models in SysML/KAOS are represented using ontologies. These ontologies are expressed using the SysML/KAOS domain modeling language.

Figure 2 is an excerpt from the metamodel associated with the SysML/KAOS domain modeling language.

2.1. Description

Each domain model is associated with a level of refinement of the SysML/KAOS goal diagram and is likely to have as its parent, through the *parentDomainModel* association, another domain model. *Concepts* (instances of Concept) designate collections of *individuals* (instances of Individual) with common properties. A concept can be declared *variable* (*isVariable=TRUE*) when the set of its individuals can be updated by adding or deleting individuals. Otherwise, it is considered to be *constant* (*isVariable=FALSE*). In addition, a concept can be an enumeration (*isEnumeration=TRUE*) if all its individuals are defined within the domain model. It should be noted that an individual can be *variable*

Email addresses: `Steve.Jeffrey.Tueno.Fotso@USherbrooke.ca`, `Marc.Frappier@USherbrooke.ca` (Steve Jeffrey Tueno Fotso and Marc Frappier), `laleau@u-pec.fr` (Régine Laleau), `amel.mammar@telecom-sudparis.eu` (Amel Mammar)

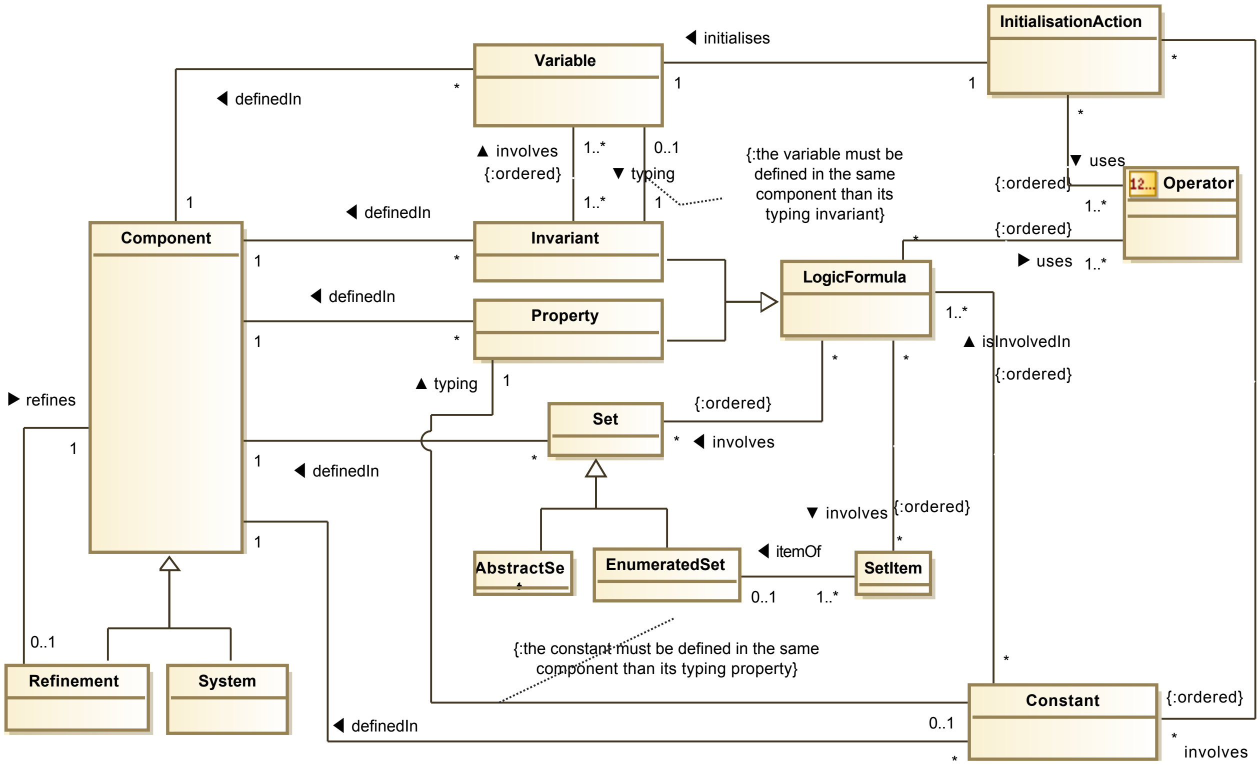

Figure 1: Metamodel of the *B System* specification language

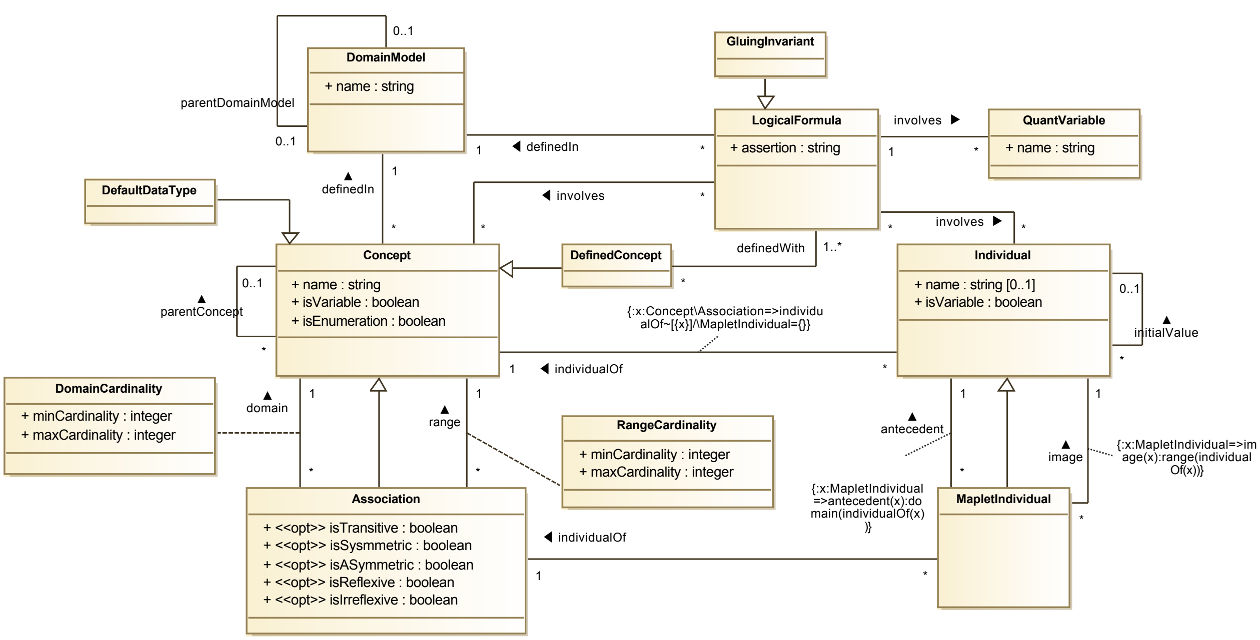

Figure 2: Excerpt from the metamodel associated with the SysML/KAOS domain modeling language

(*isVariable=TRUE*) if it is introduced to represent a system state variable: it can represent different individuals at different system states. Otherwise, it is *constant* (*isVariable=FALSE*).

Associations (instances of Association) are concepts used to capture links between concepts. *Maplet individuals* (instances of MapletIndividual) capture associations between individuals through associations. Each maplet individual references its antecedent and its image. The variability of an association is related to the ability to add or remove maplets. Each *domain cardinality* (instance of DomainCardinality) makes it possible to define, for an association `re`, the minimum and maximum limits of the number of individuals of the domain of `re` that can be put in relation with one individual of the range of `re`. In addition, the *range cardinality* (instance of RangeCardinality)) of `re` is used to define similar bounds for the number of individuals of the range of `re`.

Logical formulas (instances of LogicalFormula) are used to represent constraints between different elements of the domain model. *Gluing invariants* (instances of GluingInvariant), specialisations of predicates, are used to represent links between data defined within a domain model and those appearing in more abstract domain models, transitively linked to it through the *parent* association. Gluing invariants are extremely important because they capture relationships between abstract and concrete data during refinement and are used to discharge proof obligations. *Defined concepts* (instances of DefinedConcept) are concepts built on existing elements of the domain model using logical formulas.

2.2. Additional Constraints

- $x \in \text{Concept} \setminus \text{Association} \Rightarrow individualOf^{-1}[\{x\}] \cap MapletIndividual = \emptyset$
- $x \in MapletIndividual \Rightarrow antecedent(x) \in domain(individualOf(x))$
- $ind \in \text{Individual} \setminus \text{MapletIndividual} \Rightarrow ind \in dom(Individual\_name)$
- $ind \in \text{Individual} \setminus dom(Individual\_name) \Rightarrow Individual\_isVariable(ind) = FALSE$
- $ind \in \text{MapletIndividual} \Rightarrow (antecedent(ind) \in dom(Individual\_name) \wedge image(ind) \in dom(Individual\_name))$
- $x \in \text{Concept} \setminus \text{Association} \wedge x \notin dom(parentConcept) \Rightarrow Concept\_isVariable(x) = FALSE$
- $x \in \text{Concept} \wedge Concept\_isEnumeration(x) = TRUE \Rightarrow Concept\_isVariable(x) = FALSE$
- $(ind \in \text{MapletIndividual} \wedge Individual\_isVariable(ind) = FALSE) \Rightarrow (Individual\_isVariable(antecedent(ind)) = FALSE \wedge Individual\_isVariable(image(ind)) = FALSE)$
- $(x \in \text{Association} \wedge Concept\_isVariable(x) = FALSE) \Rightarrow (Concept\_isVariable(domain(x)) = FALSE \wedge Concept\_isVariable(range(x)) = FALSE)$

3. Translation Rules from Domain Models to B System Specifications

In the following, we describe a set of rules that allow to obtain a formal specification from domain models associated with refinement levels of a SysML/KAOS goal model.

Table 1 gives the translation rules. It should be noted that *o_x* designates the result of the translation of *x*. In addition, when used, qualifier *abstract* denotes "without parent".

Table 1: The translation rules

		Domain Model		B System	
	Translation Of	**Element**	**Constraint**	**Element**	**Constraint**
1	**Abstract domain model**	DM	$DM \in$ DomainModel $DM \notin dom(parentDomainModel)$	o_DM	$o\_DM \in$ System
2	**Domain model with parent**	DM PDM	$\{DM, PDM\} \subseteq$ DomainModel $PDM = parentDomainModel(DM)$ $o\_PDM \in Component$	o_DM	$o\_DM \in$ Refinement o_DM refines o_PDM
3	**Abstract concept that is not an enumeration**	CO	$CO \in$ Concept $\setminus$ (Association $\cap$ DefinedConcept $\cap$ DefaultDataType) $CO \notin dom(parentConcept)$ $isEnumeration(CO) = FALSE$	o_CO	$o\_CO \in$ AbstractSet
4	**Abstract concept that is an enumeration**	CO $(I_j)_{j \in 1..n}$	$CO \in$ Concept $\setminus$ (Association $\cap$ DefinedConcept $\cap$ DefaultDataType) $CO \notin dom(parentConcept)$ $isEnumeration(CO) = TRUE$ $\forall j \in 1..n, I_j \in$ Individual $\setminus$ MapletIndividual $\wedge$ $individualOf(I_j) = CO \wedge Individual\_isVariable(I_j) = FALSE$	o_CO $(o\_I_j)_{j \in 1..n}$	$o\_CO \in$ EnumeratedSet $\forall j \in 1..n, o\_I_j \in$ SetItem $\wedge$ $itemOf(o\_I_j) = o\_CO$
5	**Concept with constant parent**	CO PCO	$\{CO, PCO\} \subseteq$ Concept $parentConcept(CO) = PCO$ $o\_PCO \in$ Set $\cup$ Constant	o_CO	**IF** $Concept\_isVariable(CO) = FALSE$ **THEN** $o\_CO \in$ Constant **ELSE** $o\_CO \in$ Variable **LogicFormula:** $o\_CO \subseteq o\_PCO$

6	**Constant concept with variable parent**	CO PCO PPCO	$\{CO, PCO, PPCO\} \subseteq$ Concept $Concept\_isVariable(CO) = FALSE$ $parentConcept(CO) = PCO$ $o\_PCO \in$ Variable $PPCO \in (closure1(parentConcept))[\{PCO\}]$[1] $o\_PPCO \in$ Set $\cup$ Constant	o_CO	$o\_CO \in$ Constant **Property:** $o\_CO \subseteq o\_PPCO$ **Invariant:** $o\_CO \subseteq o\_PCO$
7	**Variable concept with variable parent**	CO PCO	$\{CO, PCO\} \subseteq$ Concept $Concept\_isVariable(CO) = TRUE$ $parentConcept(CO) = PCO$ $o\_PCO \in$ Variable	o_CO	$o\_CO \in$ Variable **Invariant:** $o\_CO \subseteq o\_PCO$
8	**Enumerated concept with parent**	CO $(I_j)_{j \in 1..n}$	$CO \in dom$(parentConcept) $isEnumeration(CO) = TRUE$ $\forall j \in 1..n, I_j \in$ Individual $\wedge$ $individualOf(I_j) = CO \wedge$ $Individual\_isVariable(I_j) = FALSE$ $o\_CO \in$ Constant[2] $\forall j \in 1..n, o\_I_j \in o\_CO$		**Property:** $o\_CO = (o\_I_j)_{j \in 1..n}$
9	**Association or defined concept without parent**	CO	$CO \in$ (DefinedConcept $\cup$ Association) $CO \notin dom(parentConcept)$	o_CO	**IF** $Concept\_isVariable(CO) = FALSE$ **THEN** $o\_CO \in$ Constant **ELSE** $o\_CO \in$ Variable
10	**Association**	AS CO1 CO2 da di ra ri	$\{CO1, CO2\} \subseteq$ Concept $AS \in$ Association $CO1 = domain(AS)$ $CO2 = range(AS)$ $Relation\_DomainCardinality\_maxCardinality(RE) = da$ $Relation\_DomainCardinality\_minCardinality(RE) = di$ $Relation\_RangeCardinality\_maxCardinality(RE) = ra$ $Relation\_RangeCardinality\_minCardinality(RE) = ri$ $o\_AS \in$ Constant $\cup$ Variable $\{o\_CO1, o\_CO2\} \subseteq$ (Set $\cup$ Constant $\cup$ Variable)		**IF** $\{ra, ri, da, di\} = \{1\}$ **THEN LogicFormula:** $o\_RE \in o\_CO1 \rightarrowtail\!\!\!\twoheadrightarrow o\_CO2$ **ELSE IF** $\{ra, ri, da\} = \{1\}$ **THEN LogicFormula:** $o\_RE \in o\_CO1 \rightarrowtail o\_CO2$ **ELSE IF** $\{ra, ri, di\} = \{1\}$ **THEN LogicFormula:** $o\_RE \in o\_CO1 \twoheadrightarrow o\_CO2$ **ELSE IF** $\{ra, di\} = \{1\}$ **THEN LogicFormula:** $o\_RE \in o\_CO1 \mathrel{+\!\!\!\!\twoheadrightarrow} o\_CO2$ **ELSE IF** $\{ra, da\} = \{1\}$ **THEN LogicFormula:** $o\_RE \in o\_CO1 \mathrel{\rightarrowtail\!\!\!\!\!+} o\_CO2$ **ELSE IF** $\{ra, ri\} = \{1\}$ **THEN LogicFormula:** $o\_RE \in o\_CO1 \longrightarrow o\_CO2$ **ELSE IF** $ra = 1$ **THEN LogicFormula:** $o\_RE \in o\_CO1 \mathrel{+\!\!\!\!\rightarrow} o\_CO2$ **ELSE** **LogicFormula:** $o\_RE \in o\_CO1 \leftrightarrow o\_CO2$ $\wedge \forall x.(x \in CO2 \Rightarrow card(o\_RE^{-1}[\{x\}]) \in di..da)$ $\wedge \forall x.(x \in CO1 \Rightarrow card(o\_RE[\{x\}]) \in ri..ra)$
11	**Individual of a constant concept that is not an abstract enumeration**	Ind CO	$Ind \in$ Individual \ MapletIndividual $CO = individualOf(Ind)$ $o\_CO \in$ AbstractSet $\cup$ Constant	o_Ind	**IF** $Individual\_isVariable(Ind) = TRUE$ **THEN** $o\_Ind \in$ Variable **ELSE** $o\_Ind \in$ Constant **LogicFormula:** $o\_Ind \in o\_CO$
12	**Constant individual of a variable concept**	Ind CO PPCO	$Ind \in$ Individual \ MapletIndividual $Individual\_isVariable(Ind) = FALSE$ $CO = individualOf(Ind)$ $o\_CO \in$ Variable $PPCO \in$ Concept $PPCO \in (closure1(parentConcept))[\{CO\}]$ $o\_PPCO \in$ Set $\cup$ Constant	o_Ind	$o\_Ind \in$ Constant **Property:** $o\_Ind \in o\_PPCO$ **Invariant:** $o\_Ind \in o\_CO$
13	**Variable individual of a variable concept**	Ind CO	$Ind \in$ Individual \ MapletIndividual $Individual\_isVariable(Ind) = TRUE$ $CO = individualOf(Ind)$ $o\_CO \in$ Variable	o_Ind	$o\_Ind \in$ Variable **Invariant:** $o\_Ind \in o\_CO$
14	**Variable individual of a concept that is an abstract enumeration**	Ind CO	$Ind \in$ Individual \ MapletIndividual $Individual\_isVariable(Ind) = TRUE$ $CO = individualOf(Ind)$ $Concept\_isEnumeration(CO) = TRUE$ $CO \notin dom(parentConcept)$ $o\_CO \in$ EnumeratedSet	o_Ind	$o\_Ind \in$ Variable **Invariant:** $o\_Ind \in o\_CO$
15	**Maplet individual**	Ind AS Ant Im	$Ind \in$ MapletIndividual $AS = individualOf(Ind)$[3] $o\_AS \in$ Constant $\cup$ Variable $Ant = antecedent(Ind)$ $o\_Ant \in$ Constant $\cup$ Variable $Im = image(Ind)$ $o\_Im \in$ Constant $\cup$ Variable	o_Ind	**IF** $Ind \in dom(Individual\_name)$ **THEN** **IF** $Individual\_isVariable(Ind) = TRUE$ **THEN** $o\_Ind \in$ Variable **Invariant:** $o\_Ind = o\_Ant \mapsto o\_Im$ **ELSE** $o\_Ind \in$ Constant **Property:** $o\_Ind = o\_Ant \mapsto o\_Im$ **LogicFormula:** $o\_Ind \in o\_AS$[4] **ELSE LogicFormula:** $o\_Ant \mapsto o\_Im \in o\_AS$

[1] *closure1(parentConcept)* designates the transitive closure of relation parentConcept

[2] Every concrete enumeration is a constant

[3] *AS* must be an association

[4] Following the variability status of *o_AS* and *o_Ind*, this predicate can be a property or an invariant

16	**Variable individual initialisation**	Ind Init CO Init_ant Init_im	$Ind \in$ Individual $\cap\ dom(Individual\_name)$ $Individual\_isVariable(Ind) = TRUE$ $o\_Ind \in$ Variable $CO = individualOf(Ind)$ $o\_CO \in$ Set $\cup$ Constant $\cup$ Variable $Ind \notin dom(initialValue) \vee (initialValue(Ind) = Init \wedge ((Init \notin dom(Individual\_name) \wedge Init\_ant = antecedent(Init) \wedge Init\_im = image(Init) \wedge \{Init\_ant, Init\_im\} \subseteq$ Constant $\cup$ Variable) $\vee\ o\_Init \in$ Constant $\cup$ Variable))		**IF** $Ind \notin dom(initialValue)$ **THEN** $o\_Ind :: o\_CO$ **ELSE** **IF** $Init \notin dom(Individual\_name)$ **THEN Initialisation:** $o\_Ind := o\_Ant \mapsto o\_Im$ **ELSE Initialisation:** $o\_Ind := o\_Init$
17	**Variable concept initialisation**	CO $(I_j)_{j \in 1..n}$	$CO \in dom($Concept$)$ $isVariable(CO) = TRUE$ $\forall j \in 1..n, I_j \in$ Individual $\wedge\ individualOf(I_j) = CO \wedge Individual\_isVariable(I_j) = FALSE$ $o\_CO \in$ Variable $\forall j \in 1..n, o\_I_j \in o\_CO$		**Initialisation:** $o\_CO := (o\_I_j)_{j \in 1..n}$[5]
18	**Association transitivity**	AS	$AS \in$ Association $Association\_isTransitive(AS) = TRUE$ $o\_AS \in$ Constant $\cup$ Variable		**LogicFormula:** $(o\_AS \ ; \ o\_AS) \subseteq o\_AS$
19	**Association symmetry**	AS	$AS \in$ Association $Association\_isSymmetric(AS) = TRUE$ $o\_AS \in$ Constant $\cup$ Variable		**LogicFormula:** $o\_AS^{-1} = o\_AS$
20	**Association asymmetry**	AS CO	$AS \in$ Association $Association\_isSymmetric(AS) = TRUE$ $o\_AS \in$ Constant $\cup$ Variable $domain(AS) = CO$ $o\_CO \in$ Set $\cup$ Constant $\cup$ Variable		**LogicFormula:** $(o\_AS^{-1} \cap o\_AS) \subseteq id(o\_CO)$
21	**Association reflexivity**	AS CO	$AS \in$ Association $Association\_isReflexive(AS) = TRUE$ $o\_AS \in$ Constant $\cup$ Variable $domain(AS) = CO$ $o\_CO \in$ Set $\cup$ Constant $\cup$ Variable		**LogicFormula:** $id(o\_CO) \subseteq o\_AS$
22	**Association irreflexivity**	AS CO	$AS \in$ Association $Association\_isIrreflexive(AS) = TRUE$ $o\_AS \in$ Constant $\cup$ Variable $domain(AS) = CO$ $o\_CO \in$ Set $\cup$ Constant $\cup$ Variable		**LogicFormula:** $id(o\_CO) \cap o\_AS = \emptyset$

Each predicate is translated with the definition of a *B System* logic formula corresponding to its assertion. Since both languages use first-order logic notations, the translation is limited to a syntactic rewriting.

Acknowledgment

This work is carried out within the framework of the *FORMOSE* project [2] funded by the French National Research Agency (ANR). It is also partly supported by the Natural Sciences and Engineering Research Council of Canada (NSERC).

[1] J. Abrial, Modeling in Event-B - System and Software Engineering, Cambridge University Press, 2010.

[2] ANR-14-CE28-0009, Formose ANR project (2017).
URL `http://formose.lacl.fr/`

[3] ClearSy, Atelier B: B System (2014).
URL `http://clearsy.com/`

[5] If $\exists j \in 1..n. I_j \notin dom(Individual\_name)$ then o_I_j must be replaced by $o\_I_j\_Ant \mapsto o\_I_j\_Im$ as in the previous rule