Trade-off Analysis for SysML Models Using Decision Points and CSPs

Patrick Leserf
ESTACA'Lab
F-53000 Laval, France
patrick.leserf@estaca.fr

Pierre de Saqui-Sannes
ISAE-SUPAERO
Université de Toulouse
F-31055 Toulouse, France
pdss@isae-supaero.fr

Jérôme Hugues
ISAE-SUPAERO
Université de Toulouse
F-31055 Toulouse, France
jerome.hugues@isae-supaero.fr

ABSTRACT

The expected benefits of Model-Based System Engineering (MBSE) include assistance to the system designer in finding the set of optimal architectures and making trade-off analysis. Design objectives such as cost, performance and reliability are often conflicting. The SysML-based method OOSEM and the ARCADIA method focus on the design and analysis of one alternative of the system. They freeze the topology and the execution platform before optimization starts. Further, their limitation quickly appears when a large number of alternatives must be evaluated. The paper avoids these problems and improves trade-off analysis in a MBSE approach by combining the SysML modeling language and so-called "decision points". An enhanced SysML model with decision points shows up alternatives for component redundancy, and instance selection and allocation. The same SysML model is extended with constraints and objective functions using an optimization context and parametric diagrams. Then a representation of a constraint satisfaction multi-criteria objective problem (CSMOP) is generated and solved with a combination of solvers. A demonstrator implements the proposed approach into an Eclipse plug-in; it uses the Papyrus and CSP solvers, both are open-source tools. A case study illustrates the methodology: a mission controller for an Unmanned Aerial Vehicle (UAV) that includes a stereoscopic camera sensor module.

Keywords: MBSE, Optimization, SysML, CSP, Papyrus, System engineering, Optimal architecture design, Decision points.

1. INTRODUCTION

According to INCOSE [1], Model-based systems engineering (MBSE) is the formalized application of modeling to support system engineering activities, from requirements to validation. These activities have traditionally been performed using document-based approaches. The expected benefits of MBSE include better specification and design quality, reuse of design artifacts, and a coherent model of the system to be developed.

Selecting a modeling language is a key issue for MBSE. Originally, mathematical formalisms (e.g. [2]) were introduced, allowing analysis and optimization by means of specific tools. Over the past decade, joint efforts of OMG and INCOSE have lead to the standardization of SysML [3], a modeling language that addresses important issues such as requirements, architecture and behaviors.

The paper addresses one MBSE activity within a SysML context: trade-off analysis among alternatives for the system model in order to meet design objectives, such as cost, performance, reliability and other inputs from the stakeholders' needs. These needs are often conflicting, and the goal of trade-off analysis is to provide a balanced solution [4].

To find a balanced solution, several methods are available to specify, design, and verify the system to build. The Object-Oriented System Engineering Method (OOSEM) from OMG [4], is the only one using SysML.

The specification and design steps of OOSEM include two highly important activities [4]:

1. Synthesis of alternative variants by structuring the system, and
2. Evaluation of variants associated with trade-off analysis so as to determine a set of optimal solutions.

A designer who synthesizes alternative variants needs to minimize objectives such as cost, performance and failure rate. Examples of objectives include cost and performance, redundancy level for failure rate, and allocation for performance. [5] and [6] rely these design decisions on a pure optimization problem and separate them from design representation. Conversely, the paper discusses a MBSE approach based on SysML language [3].

With OOSEM and SysML 1.5, OMG introduced stereotypes to allow trade-off analysis in the form of objective functions and measure of effectiveness (moe). For each variant of the system, a «moe» stereotype models the values to be optimized, and an external tool computes the objective functions values. A component variant is explicitly defined by inheritance from generic components, and only a limited number of variants shall be considered. Our approach allows modeling a large number of variants by using "decision

points". The designer uses them to model variation in component instance choices, redundancy level or allocation. The notion of decision points is close to that of variability [7], but remains specific to system engineering decisions and to their combinations.

In terms of drawback, OOSEM/SysML aggregates the different objectives into a single one called the "utility function". In the paper, a new approach suggests the best configurations to the designer, and finds the Pareto-optimal solutions [8] that have the lowest (or equivalently low) values for all objective functions.

For the designer, the benefits of our approach is threefold. First, it allows to model a large number of alternatives, without having to define them explicitly in detail. Second, it allows a real optimization process from the model, instead of a simple analysis of the different alternatives of the system. Third, we propose a Pareto Front analysis of the optimal solutions, instead of a global ranking based on the weighted sum of the different objective. This allows a better decision process, with more degree of freedom and fewer hidden solutions. At the end, the solution selected by the decision maker is highlighted in the model, which allows round-trip optimization even if it is done manually for the moment.

In our approach, we provide to the designer a way to model alternatives (the decision points) and objectives. Then the proposed algorithms generate a constraint satisfaction and multi-criteria objective problem (CSMOP) representation from the (SysML) model. The designer can solve the CSMOP problem and select solutions. At the end, a proof of concept is achieved by interfacing the Papyrus SysML tool with several solvers.

Figure 1 depicts a corollary contribution in the form of a three-step method:

❶ SysML Modeling for optimization (cf. sections 3 and 4). An initial SysML model describes a system without alternative. New stereotypes extend the model for optimization purposes. A SysML parametric diagram models a "MDO context" optimization context and contains the model variants (decision points). The solver is selected, and objective functions are defined using the solver language.

❷ Model transformation. The SysML model produced by the first step is transformed into a description of a CSMOP problem. The CSMOP description defines variables by their domains, global constraints and objective functions.

❸ Best solutions generation. The optimal solutions are calculated with different solvers such as CHOCO [9] or PyOpt [10], depending on the kind of decision points, corresponding to discrete or continuous problems. The designer can select the solutions that better fit the requirements in term of power, performance or any other type of metric. The selected design is used for domain-specific optimizations such as scheduling analysis.

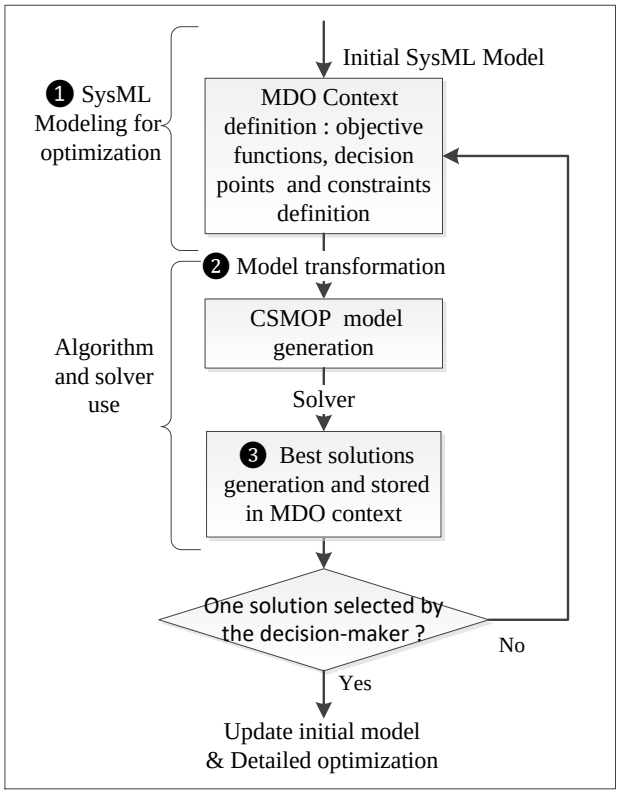

Figure 1 "Methodology Overview"

The paper is organized as follows. Section 2 gives the background for SysML modeling, variants modeling and optimization. Section 3 addresses meta-model for decision points, and transformation algorithms. Section 4 instantiates in SysML. Section 5 adds a plug-in to Papyrus and applies the proposed approach to a UAV modeled. Section 6 surveys related work. Section 7 concludes the paper and outlines future work.

2. BACKGROUND

2.1 SysML

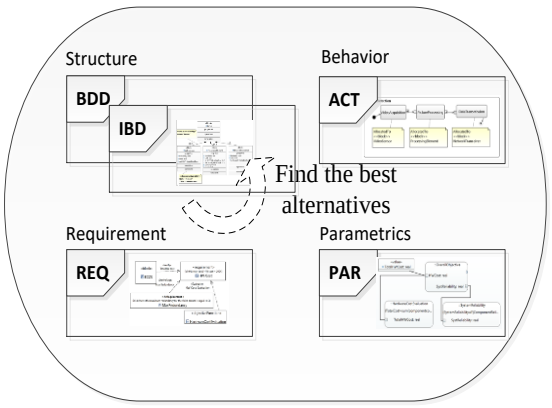

Figure 2 "SysML diagrams"

A system model in SysML is made up of a set of diagrams categorized into four groups (Figure 2). Requirement

diagrams (REQ) describe requirements. Activity diagrams (ACT) represent the behavior of the system. Block Definition Diagrams (BDD) and Internal Block Diagrams (IBD) describe the architecture of the system. Finally, parametric diagrams represent constraints on property values such as $U = RI$, used to support engineering analyses.

Figure 1 also links optimization to the so-categorized diagrams. For example, the *Allocate* relationship links elements from different SysML diagrams. *Allocate* modifies the allocated element, creating problems when different allocation patterns are needed. The *Assign* stereotype is an alternative representation for the allocation, based on semantic neutral UML::Comment. The *Assign* stereotype can be used either to specify possible allocation (allowing to perform optimization) or to specify an actual allocation in the system, depending on the context. To represent variable allocation, we can create a new stereotype deriving from *Allocate*, corresponding to the concept we need for trade-off analysis.

2.2 MBSE Method and trade-off analysis

When the OMG consortium standardized the first version of SysML in [11], the OOSEM method was proposed by [2]. The OOSEM activities produce artifacts represented by SysML diagrams and stereotypes. OOSEM [4] is a top-down, scenario-driven process that supports the analysis, specification, design and verification of systems. During the design process, trade-off analysis is a major activity of OOSEM.

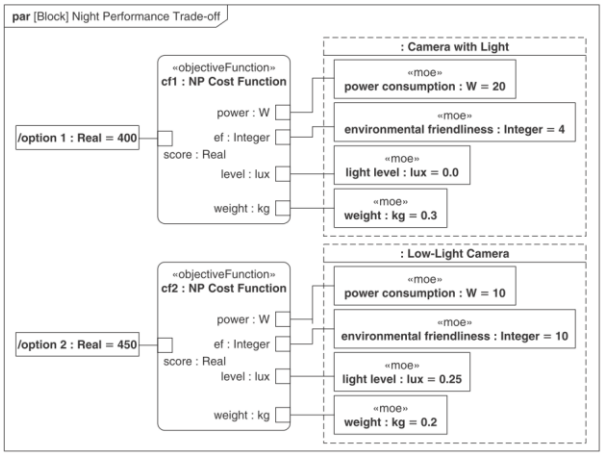

Figure 3 "Trade-off analysis with OOSEM"

To perform trade-off analysis, OOSEM recommends building an analysis context, with a Parametric Diagram (PAR) and a Block Definition Diagram (BDD). These diagrams contain functions tagged with the «ObjectiveFunction» stereotype. A global performance value is obtained with a weighed sum, and calculated with an external tool. To compare different variants for the system, each variant is modeled with block values tagged with «moe» stereotypes. Figure 3 (from [4]) presents a PAR diagram with two variants for a camera: "with light" and "low light". A global performance value is calculated from four objectives including weigh and light level. The specialization mechanism is used to model each variant from SysML blocks.

2.3 Pareto Frontier

Figure 3 depicts a cost function where the designer is inclined to weigh each objective, according to their importance. The main drawback of this method stems from the linear approximation of the global performance function. The concept of multi-objective optimization or Pareto optimality addresses these problems [8]. It describes a multi-objective optimization problem by:

$$min\ \boldsymbol{f}(x) = [f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \dots f_n(\boldsymbol{x})\ ]\ with\ \boldsymbol{x} \in S$$

Above, $\boldsymbol{f}$ is the objective function vector and S the set of solutions. As far as system design is concerned, the objective functions and constraints can be linear (such as the cost) or nonlinear (such as failure rate). For a minimization problem, an alternative named A dominates another one named B if and only if:

$$\begin{cases} \forall i \in \{1..n\} & f_i(\boldsymbol{a}) \leq f_i(\boldsymbol{b}) \\ \exists i \in \{1..n\} & f_i(\boldsymbol{a}) < f_i(\boldsymbol{b}) \end{cases}$$

We consider as solution the set of non-dominated assignments. The Pareto frontier in Figure 4 consists of all alternatives that are not dominated by another one. The Pareto frontier is a powerful help for the designer, compared to the weighted sum approach.

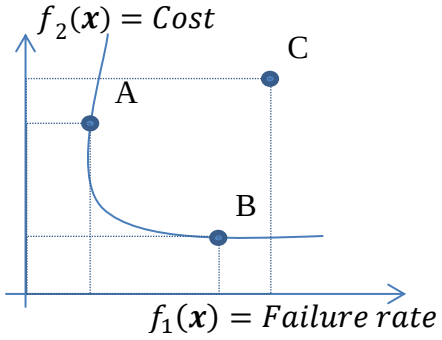

Figure 4 "Pareto frontier"

2.4 Classification of decisions problems

Figure 3 depicts a trade-off analysis where the designer selects one component instance (a camera) from a list of off the shelf components. This is an instance decision problem, a decision problem largely encountered in system engineering.

Another common decision problem deals with attributes values of the blocks. It corresponds to a value decision problem, where value types can be integer (discrete problem), real (continuous problem) or both (mixed problem). The designer wants to find the optimal combination of attributes to minimize (or maximize) several objectives.

For a designer using MBSE, instances and attributes values are not the only degree of freedom for the system. The structure of the system has to be considered too, especially at early phase of design. The typical representation of this problem is the Redundancy and Allocation Problem (RAP) discussed in [6] and [12]. The RAP problem deals with component selection, for cost and reliability optimization at system level. It is formalized as an optimization problem. It is not connected to any MBSE approach. In the RAP problem, the connection topology is fixed as a serial-parallel model.

The last degree of freedom generally studied in system design trade-off is the allocation of sources elements onto target elements. This happens with embedded system design, where the applications elements are allocated to the Processing Elements (PE) of the HW platform.

In [13], the authors compare software architecture optimization methods, few of them using SysML or UML. Concerning the degrees of freedom, component selection/duplication (instance and redundancy) represents 40% of the approaches and allocation represents 33%.

2.5 CSMOP problems for trade-off analysis

Trade-off analysis can be obtained by resolving a multi-objectives optimization problem (CSMOP) including constraints. The following definition is applicable:

Definition 1. A Constraint Satisfaction MultiObjective Optimization Problem (CSMOP) is a quadruple $\{\boldsymbol{x}, D, C, \boldsymbol{f}\}$ made up of

- An array of decisions variables $\boldsymbol{x} = [x_1, x_2, .. x_n]$,
- A set of n domains $D = \{D_1, D_2, .. D_n\}$ with $x_i \in D_i$,
- A set of m constraints $C = \{C_1, C_2, .. C_m\}$ where C_i is a Boolean function involving a sequence of variables $X(C_i) = \{x_{i1}, x_{i2}, .. x_{im}\}$ called its scope,
- An array of functions $\boldsymbol{f}(\boldsymbol{x}) = [\, f_1(\boldsymbol{x}), f_2 \ldots f_n(\boldsymbol{x})) \,]$ where $f_i(\boldsymbol{x})$ is an objective function, which maps every solution to a numerical value.

The set of domain D refers to either **continuous domain** when $D_i \subset \mathbb{R}$ or **discrete domain** when $D_i \subset \mathbb{N}$.

In [14], the authors propose to iteratively resolve a CSMOP problem $M_0 = \{\boldsymbol{x}, D, C, \boldsymbol{f}\}$ and to derive from it a basic Constraint Satisfaction Problem (CSP) $N_0 = \{\boldsymbol{x}, D, C\}$. Then N_1 is obtained from N_0 and M_0 by adding a new constraint $c_i = f_i(\boldsymbol{x})$. An initial solution S_1 for the N_1 problem is found using a branch and bound algorithm. A new constraint $c_i < f_i(S)$ is then added to N_1 to obtain N_2 and a new solution S_2 to N_2can be found. These steps are repeated until no solution is found. The last solution found is the optimal one for $f_i(\boldsymbol{x})$.

This approach relies on CSP problem resolution. CSP are widely used in combinatorial and optimization problems but also in continuous domain, as needed for trade-off analysis in MBSE. It has a great advantage: its declarative nature allows the constraints to be expressed in a natural way, and existing algorithms are particularly effective for reducing the size of the search space. Several solvers exist such as CHOCO [9] [15] for discrete variables, and PyOpt [10] for continuous domain.

2.6 Variant modeling

During the trade-off analysis, the designer of a complex system has to evaluate a large number of alternatives, and a specific approach is needed.

With variant modeling [16], one specifies design alternatives by explicitly modeling them in a single model, and annotating them using variation points. Variant modeling is often associated with software product lines and feature models [17], with the intention to create many variants of a product. A feature diagram is hierarchically organized, starting with a feature node at the root position. Another technique is to use a separate variability language, such as *CVL* [18]. For optimization, it is essential for the designer to clearly identify the SysML elements subject to variability, rather than to define them in a separated language or diagram. This is why the paper uses extensions of SysML to define variability for optimization purposes. However, relying optimization on "decision points" remains compatible with the CVL concepts, and is a subset of CVL features.

2.7 Summary

OOSEM/SysML proposes a weighted-sum approach to perform trade-off analysis, which is fully convenient when the designer evaluates only few variants of the system model. Variant modeling and feature models can model a large number of variants, but the elements submitted to variation are not clearly identified. For the system designer, four different degrees of freedom shall be considered: instance, redundancy, values and allocation. However, trade-off analysis can be obtained directly by resolving a multi-objectives optimization problem (CSMOP) including constraints, using a CSP solver.

Instead of formulating the CSMOP problem directly, there is a need for the system designer to generate and solve CSMOP problem from the SysML model. To model the degrees of freedom, we propose to use "decision points" to represent instance choice, redundancy and allocation. The augmented model is transformed into a CSMOP problem, and a solver provides the Pareto frontier.

3. Decision points, Constraints and Context

Previous section shows how it is important for the system designer to model four degrees of freedom: instance and redundancy, values, and allocation. This section represents the corresponding decision points with a meta-model and sketches algorithms creating variables for a CSMOP problem.

3.1 Decision points for CSMOP

3.1.1 Redundancy and instance choice

The meta-model describing decision points is not restricted to SysML language and may be reused for another MBSE language such as AADL language [19]. The MBSE

language we consider supports the concept of class and composite relations between these classes. Each class has a set of typed attributes. The system to be optimized is a set of classes and composite relations. A composite relation associates a whole class with a set of parts, with a given multiplicity. A set of instances is also associated with a class, because system analysis requires taking both instances and their attributes values into account.

The decision points (*DP*) for redundancy and instance choice are represented in Figure 5. The "InstanceDecision" *dpi* is associated with a class and a set of instances $IB = \{ib_1, ib_2 \dots ib_m\}$, and is used to represent a component choice. The "StructureDecision" *dps* is linked to a composite relationship to represent a variable redundancy $r \in \{1 \dots n\}$ of a class in the system.

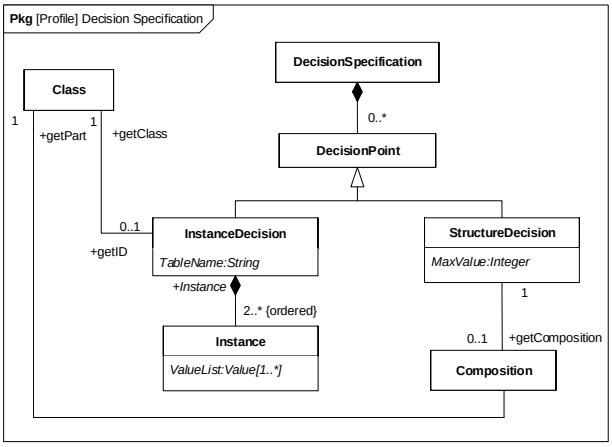

Figure 5: Decision points for instance and structure

From the decision points, it is possible to create several variables and domains for the definition of a CSMOP problem. We consider the following alternatives for *dpi* and *dps*:

1. The class C has only one decision point *dpi*. For this single degree of freedom, one bounded integer variable $x \in [1..\mathrm{m}]$ identifies one instance ib_i
2. The class C has one decision point *dps*. One bounded integer variable $y \in [1..\mathrm{n}]$ identifies the redundancy level of the class.
3. The class C has both decision points *dpi* and *dps*. For this combined configuration, we create a matrix of Boolean variables $x_{ij} \in \{0,1\}\ i \in \{1..\mathrm{n}\}, j \in \{1..\mathrm{m}\}$ where:

$$x_{ij} = \begin{cases} 1\ if\ ib_j\ is\ used\ in\ i\ position\ of\ the\ redundancy \\ 0\ otherwise \end{cases}$$

Equation 1 "dps and dpi variables"

From a given model including two sets of decision points *DPS* and *DPI*, a three-steps algorithm generates the variables:

Step 1: Check the consistency of the model. Each $dp \in \{DPI \cup DPS\}$ has to be linked to a class of the model. Otherwise the designer has to change the model.

Step 2: For each *dps* linked to a Class C, search if a *dpi* is attached to C. If so then create a Boolean matrix x_{ij} and remove *dpi* from *DPI*. Otherwise create a bounded integer variable for *dps*.

Step 3: Create a bounded integer variable for each $dpi \in DPI$

3.1.2 Allocation problem

For the allocation problem, the target MBSE shall implement a mechanism allowing connecting elements at different levels of abstraction, or with different types such as software and hardware. It is possible to allocate an activity to a resource for execution, or a small physical component to a bigger one with a given volume. In order to find the set of optimal allocations, we use the variable allocation depicted by Figure 6.

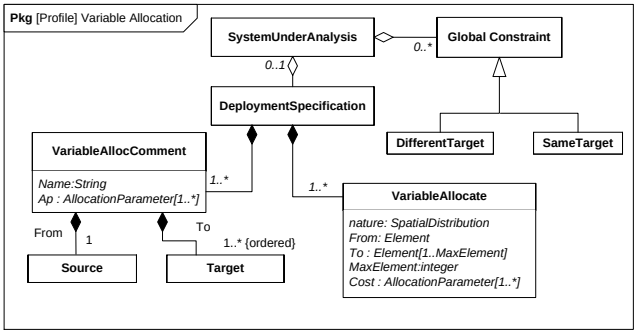

Figure 6 "variable allocation for optimization"

The system under analysis must be characterized by a deployment specification containing all possible allocations of the system. Each allocation can be represented by "VariableAllocateComment" when the source and targets are represented in the same diagram. "VariableAllocate" is used otherwise. In both cases a unique source element can be allocated to a set of target elements with specific parameters, such as a cost for each allocation. Specific constraints for allocation can be added, to specify a target for a set of source elements.

The deployment specification is checked with the verification of each variable allocation. A valid variable allocation shall contain a unique source and a set of at least two target elements.

From the model with variable allocation, it is possible to create variables $X = \{x_1, x_2, .. x_n\}$ and domains $D = \{D_1, D_2, .. D_n\}$, for the definition of an optimization problem. X and D are obtained from $S = \{s_1, s_2, .. s_n\}$, the set of source elements and from $T = \{t_1, t_2, .. t_m\}$ the set of target elements of variable allocation.

First formulation with $D_i \subset \mathbb{N}$

$$\forall s_i \in S, \forall t_j \in T \quad x_i = j \Leftrightarrow s_i\ is\ allocated\ to\ t_j$$

Equation 2 "integer formulation for allocation"

Second formulation with $D_i = \{0,1\}$

$$\forall s_i \in S, t_j \in T \quad x_{ij} = \begin{cases} 1 \text{ if } s_i \text{ is allocated to } t_j \\ 0 \text{ otherwise} \end{cases}$$

Equation 3 "Boolean formulation for allocation"

Second formulation has a drawback: it creates a matrix of Boolean instead of a scalar. But it offers simpler expressions for constraints and objective functions. From a given deployment specification, the below algorithm generates the variables of the second formulation:

Step 1: Create Boolean matrix x_{nm} from the source list and the target list TA_1.

Step 2: For each array x_i, add a constraint to restrict the allocation of one source to only one target.

Step 3: For each Variable allocation v of the model and x_i, obtain the set TA_2 of possible targets. Then for each $t \in TA_1$ if $t \notin TA_2$ then $x_{ij} = 0$.

3.2 Constraints

In MBSE, several constraints limit the number of variants during the search for optimal solutions. Figure 7 identifies a set of constraints for system optimization and combines them with decision points.

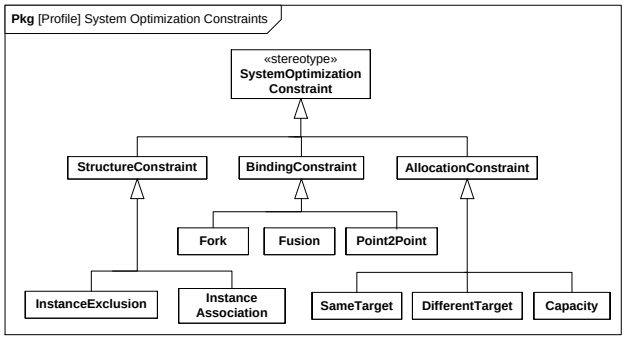

Figure 7 "Constraints for system optimization"

The "StructureConstraints" are used with structure and instance decision points. When several instances of two classes C1 and C2 are constrained with "InstanceAssociation", these instances must be associated in the real system. If we consider the variables defined by Equation 1, we can add the following constraint to the optimization problem, associating instance *j* of class *A* and instance *k* of class *B*:

$$\sum_{i=1}^{n} a_{ij} \leq \sum_{i=1}^{n} b_{ik}$$

Equation 4 Instance Association

The binding constraints are used when two classes C1 and C2 are connected through connection ports. We have constraints between the total input port number e_2 of C2 and the total output port number s_1 of C1, depending on the kind of connection: fork, fusion or point to point (Figure 8).

Fork Fusion Point to point

Figure 8 "Connection type for class ports

For a "fork" constraint, we have:

$$\sum_{i,j} b_{ij} e_2 \geq \sum_{i,j} a_{ij}\, s_1$$

And for a fusion constraint:

$$\sum_{i,j} b_{ij} e_2 \leq \sum_{i,j} a_{ij}\, s_1$$

Regarding as allocation constraints, we consider $\beta \subseteq S$ a set of source elements that shall be allocated (or not) to a target $t \in T$. The integer formulation (Equation 2) adds a global constraint to the problem:

$$AllEqual(x_i) \; with \; x_i \in \beta$$

The Boolean formulation (Equation 3) adds a set of m constraints:

$$\forall j \in \{1..m\} \; AllEqual(x_{ij}) \; with \; x_i \in \beta$$

Capacity constraints include both the capacity use, such as memory use, and the utilization factor of each target element. They can be easily expressed with the Boolean formulation. With m_j the total amount of capacity for the target t_j, m_i the resource capacity needed for resource s_i, we have:

$$\forall j \in \{1..m\} \quad \sum_{i} x_{ij} m_i \leq m_j$$

3.3 Optimization context

A *Multi Domain Optimization Context* (***MDO Context***) represents a situation that needs to be optimized in order to maximize or minimize a set of objective functions. The MDO context is a part of the system model and brings together the different elements needed to optimize the system. For the system designer, MDO context is the central point to drive optimization.

MDO Context is defined in Figure 9 and includes:

1. A reference toward the system under analysis (*SUA*), which contains the previously defined decision points, for instance, redundancy and allocation.
2. An optimization model "OptModel" containing the mathematical representation of variables. Specific Boolean or integer models can be used. The OptModel has constraints, corresponding to a mathematical expression, on the language of the solver. The constraints and variables of the OptModel can be generated from the decision points and user defined constraints, with the algorithms proposed by the paper.

3. One or several objective functions. Each function calculates one objective value to be minimized or maximized. Parameters of this function include the variables of the *OptModel* and the attributes values of the *SuA*. The *ParetoFront* is used to generate optimal solutions, according the different objectives and with the selected solver. The solver choice is a parameter of the *MDO Context*.

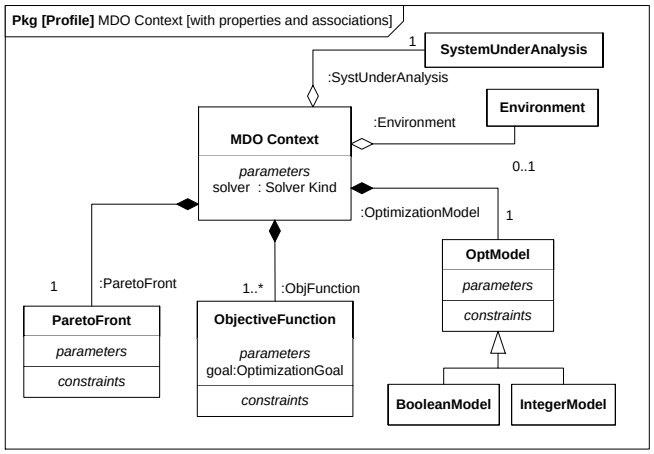

Figure 9 "MDO Context"

The environment represents the context of use, in which we want to find out optimal solutions. The environment may include static elements such as data defined by Classes, or dynamic scenarios described by SysML sequence diagrams.

Table 1 "Connection with SysML and AADL"

Generic element	SysML metaclass	AADL element
Class	Block	Device, memory, processor, bus
Composition	Block, BDD diagram	Subcomponents
Class Attribute	Property	Properties
Allocation	Allocate for behavior	Allowed processor binding property
Source elements for allocation	Action/Activity	Single thread or list or thread
Target elements for allocation	Block or part	Single processor or processor list
Class instance	Instance diagram	Instance model
Constraint	Constraint block	Annotation

3.4 Summary

The meta-model proposed in this section represents decision points, constraints and an optimization context. It supports optimization activities with any MBSE language. We assume that the target MBSE language is composed of generic elements needed for system optimization. Table 1 lists these generic elements and connects them to SysML and AADL.

4. SysML and solver integration

4.1 Stereotypes for optimization

Modeling languages such as UML or SysML are defined using metamodeling, describing the language concepts with metaclasses [20]. The stereotype is a special type of metaclass, derived from an existing UML concept or from another stereotype. To make the concepts presented in section 3 compatible with the UML and SysML semantics, Figure 10 connects decision points to SysML metaclasses and stereotypes.

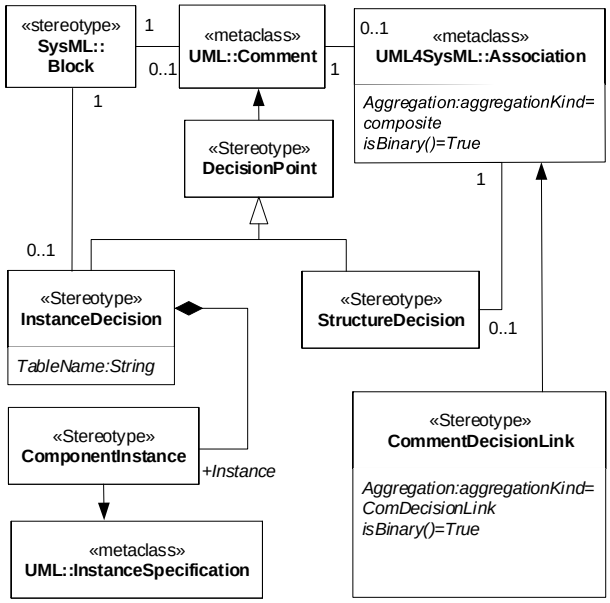

Figure 10 "Decision points with SysML"

In Figure 10, the "DecisionPoint" stereotype extends the UML::Comment metaclass. The "InstanceDecision" is connected to a SysML block, and to an instance specification from SysML. The "StructureDecision" applied to a composite association represents a variable redundancy. A specialization of association "CommentDecisionLink" connects a decision point to a block.

In Figure 12, a SysML model fragment shows a video sensor block with decision points for instance and structure. The sensor block can be duplicated or not for redundancy, and each block has two possible instances. Each instance has cost and reliability values to compute the cost and the reliability of the system.

The validation of the annotated model is proceed by checking the correct use of decision points. Instance decision point shall be connected to a block, and the structure decision point to a composite association with a minimum cardinality of two on the part side.

The optimal combination of instances and redundancy is obtained by using the Pareto front representation, modeled by the MDO context.

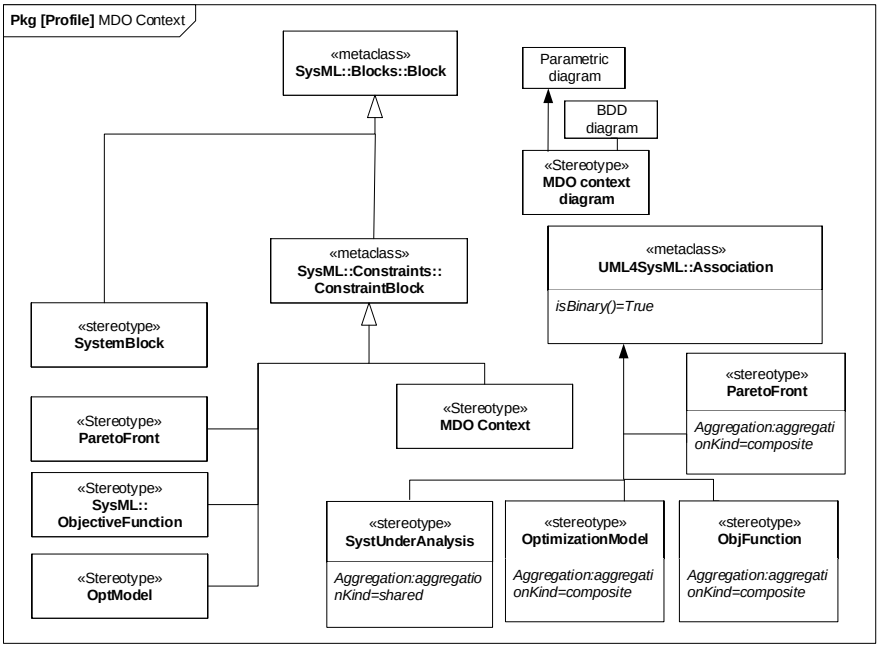

Figure 11 "MDO context with SysML"

The MDO Context integration in SysML is represented in Figure 11. SysML introduces the concept of "diagram usage" to represent a particular usage of a diagram type. A diagram is not a metaclass in UML but the concept of extending a diagram for a particular diagram usage is possible with SysML [3]. In Figure 11, the stereotype notation represents the MDO context diagram, extending the parametric diagram.

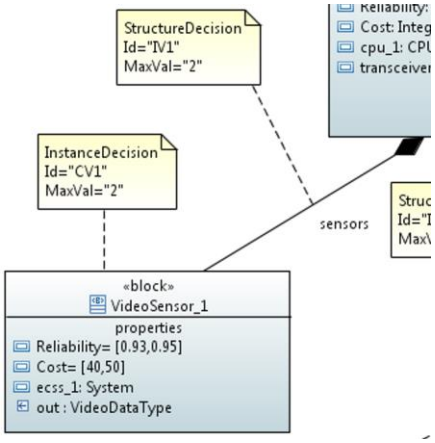

Figure 12 "SysML model with decision points"

A parametric diagram is a restricted form of internal block diagram (IBD). It contains constraint blocks representing equations. That is why the *MDO Context* refers to both BDD and parametric diagrams. In Figure 11, the "*MDO Context*" block extends the SysML stereotype "*ConstraintBlock*". This constraint block is used as a top-level block for the associated parametric diagram. This constraint block has a reference (not shown in the figure) towards the "*SystemBlock*" which is the representation of the system under analysis. The "*SystemBlock*" includes the decision points stereotypes described in previous paragraph. The "*OptModel*" block shows the representation of the decision variables given by Equation 1 "dps and dpi variables", using an integer array or Boolean matrix. Using both types requires creating two sets of variables and inserting them as parameters in the "*OptModel*" block, using constraints. The objective functions are other parts of the "MDO *Context*". Each objective function has parameters for binding inputs with the system under analysis and with the "*OptModel*" constraint block.

A key point for the problem formulation and complexity is the decision variable definition, because it directly influences the framework performances. We propose *Algorithm 1* to assist the SysML designer in this task. This algorithm creates the appropriate decision variables and constraints used by the framework, from the deployment specification contained in the SysML model. The algorithm variables are listed in Table 2.

Table 2 Algorithm variables

Name	Stereotype	Description
VA	«Deployment Specification»	The set of variable allocate elements in the input SysML model
v	«VariableAllocate»	The current variable allocate element
s	« Source»	The source element to be allocated, such as a task
TA	«TargetList»	The set of target elements allocated to s with the variable allocate

t	«Target»	The current target element, with identifier "id"
P		The ouput CSP problem to build

The algorithm goes through the deployment specification of the SysML model containing the list of "*variableAllocate*" elements depicted by Figure 6. Two encoding type can be selected. If the encoding type in the SysML model is Boolean, *Algorithm 1* is used. A Boolean matrix of decision variables matches the entire deployment specification. We start to retrieve the list of available sources and target elements from the model (line 3-6). Then for each source element of the model, we add a global constraint (line 10, *AddAtMostConstraint*) corresponding to equation (3) meaning that a source is allocated to only one target. For each variable allocation, if a source is not allocated to an existing target as given in equation (5), we set the corresponding Boolean variable to FALSE (line 15).

Table 3 Algorithm primitives

Name	Description
GetFrom(), GetTo()	Retrieve the list of sources and target elements of relation "*variableAllocate*".
CreateBoolMatrixVariable()	Create variables for P problem
AddBoolConstraint()	Add a Boolean constraint to P, with one Boolean variable,
AddAtMostConstraint()	Add at most constraint for matrix of Boolean variable,
AddScalarConstraint()	Add a scalar constraint to P, with an array of variables

After the variable creation, the constraints defined in paragraph 3.2 can be generated (line 20-26). Capacity constraint generation is shown, using a scalar constraint and a transposed Ts Boolean matrix. A *SameTarget* constraint is generated also with AddBoolEqConstraint.

Table 3 gives the list of primitives used by the algorithm. The primitives are general enough to apply to any optimization problem, but their specific implementation depends on the selected solver.

We obtain the following results for the CHOCO solver with a model including four target elements and one source element, with one variable allocation from s0 to target 0,1,3 , one capacity constraint and one *SameTarget* constraint.

4.2 Solver integration

We evaluated several CSP and optimization solvers. The goal is to integrate them in an open-source framework, such as Eclipse [21] and Papyrus [22]. We can classify the solvers in two groups. The first group contains black-box tools such as Minion [23] or RealPaver [24]. These solvers are not suitable for us, because we want to integrate them into an open-source framework.

Algorithm 1: decision variable creation from SysML model with Boolean Matrix and capacity constraints

01: ***Input***: $VA = \{Deployment\ Specification\ of\ the\ SysML\ model\}$
02: $CSP\ problem\ P \leftarrow \emptyset$
03: $SourceList\ \ S \leftarrow Model.getSourceElmts()$
04: $TargetList\ \ TA_1 \leftarrow Model.getTargetElmts()$
05: $ConstrList\ \ C\ \ \leftarrow Model.getConstraints()$
06: $int\ P_{max} = TA_1.Card()$
07: $int\ S_{max} = S.Card()$
08: $P.CreateBoolMatrixVariable(s, S_{max}, P_{max})$
10: $AddAtMostConstraints(s, S_{max})$
11: ***For*** v ***in*** VA :
12: $TargetListList\ TA_2 \leftarrow v.getTo()$
13: ***For*** t ***in*** TA_1 :
14: ***If*** $(t \notin TA_2)$
15: $P.AddBoolConstraint("s[i][j] = False")$
16: $j++;$
17: ***Endfor***
18: $i++$
19: ***Endfor***
20: $ConstrList\ \ C\ \ \leftarrow Model.getConstraints()$
21: $P.CreateBoolMatrixVariable(Ts, s)$
22: ***For*** c ***in*** C :
23: ***If*** $(c.Type = Capacity)$
24: AddScalarConstraint("≤", Ts, S, TA_1)
25: ***If*** $(c.Type = SameTarget)$
26: AddBoolEqConstraint(C, s, TA_1)
27: $k++;$
28: ***Endfor***
29: ***Return*** P

Result obtained with Algorithm 1 :

```
private static final int S_MAX=2;
private static final int T_MAX=4;
BoolVar[][] s =
VariableFactory.boolMatrix("s", S_MAX,T_MAX,
solver);
For(int i=0;i<S_MAX;i++)
   // a source s is allocated to one target
   SatFactory.addAtMostOne(s[i]);
// allocation to t2 is excluded for s0
SatFactory.addBoolEq(s[0][2],FALSE);
For(int j=0;j<T_MAX;j++) {
   // Same Target constraint for s0 and s1
   SatFactory.addBoolEq(s[0][j], s[1][j]);
   // Capacity Constraints for targets
   solver.post(ICF.scalar(Ts[j],MemorySource
,"<=",TargetMemory));
}
```

Table 4 Solver evaluation

Solver			**PyOpt**	**Labix**	**Choco**	**ECL^{i}PS^e**
Language			Python	Python	Java	ECL^{i}PS^e
Domains	Boolean		+	-	+	+
	Integer		+	+	+	+
	Real		++	-	+	+
	Set of variables		+	-	++	+
Constraints						
	Arithmetic	integer	+	+	+	+
		real		-		
	Global		-	-	++	+
Optimization			yes	no	yes	no
	Mono/Multi-objective		multi ++(genetic)	no	multi +	-
	Algorithms		++	-	+	-
Suitable for decision points			Value decision, real types	Instance, Structure	Instance, Structure, Allocation	Instance, Structure

The second group includes object libraries and functions dedicated to CSP and optimization problem, written in JAVA or Python language. This is the group of solvers we have investigated: Labix [25], PyOpt [10], CHOCO [9] and ECL^{i}PS^E [26]. The first criterion for choosing a solver is the type of variables it uses. For the previously defined decision points, we need both integer and real variables. A second important criterion is the ability to handle sets of variables and global constraints. With this feature, we easily formulate system constraints such as "AllDifferent" on a group of variables. We considered also the possibility to perform multi-objective optimization inside the solver, and the use of different algorithms (Backtracking, MAC) for CSP problem solving.

From the results presented in Table 4, we selected the Labix solver for our first experimentations with structure and instance decision problems. The PyOpt solver is suited for variables in continuous domain, which is useful for value decision problems. The CHOCO solver is similar to Labix, but with much more capabilities. CHOCO can handle global constraints, needed for allocation problems with a large exploration space. The ECL^{i}PS^E solver was not selected because the solver was coded using a declarative language similar to Prolog.

5. Case Study and tool

A UAV model serves as case study for the methodology and the algorithms implemented as a plug-in of the SysML Papyrus tool.

5.1 Autonomous UAV

Autonomous Unmanned Aerial Vehicles (UAVs), sometimes called flying robots, are being used for intelligence, surveillance, and reconnaissance missions [27]. Autonomous UAVs have an increased level of autonomy and more complex scenarios are envisioned [28]. UAVs range from remotely piloted vehicles to more sophisticated, fully autonomous UAVs. At an intermediate level for autonomous UAV, fault or event adaptive UAVs perform on-board trajectory re-planning from obstacle detection, and combine this feature with a mission realization. Communication to ground control system cannot be permanently guaranteed and on-board power is limited. Therefore, processing and decision-making are entirely done on board. The HW reliability and the system cost have to be considered first. The HW platform is made up of CMOS image sensors, processing elements and UAV interface networks (transceivers). These three components may be redundant, for safety purposes, and they are selected in a repository of instances. In the next paragraph, Our optimization approach is used to determine the optimal HW configuration, with a SysML model including decision points, and a CSMOP problem generation and solving.

5.2 Hardware redundancy and instances

The hardware system is made up of several components and described by a block definition diagram. The HW platform for the UAV system in Figure 13 contains one or two sensors, processing elements and transceivers for onboard network. Three decision points for redundancy and three for instances are respectively related to the sensor, the CPU and the Transceiver composition.

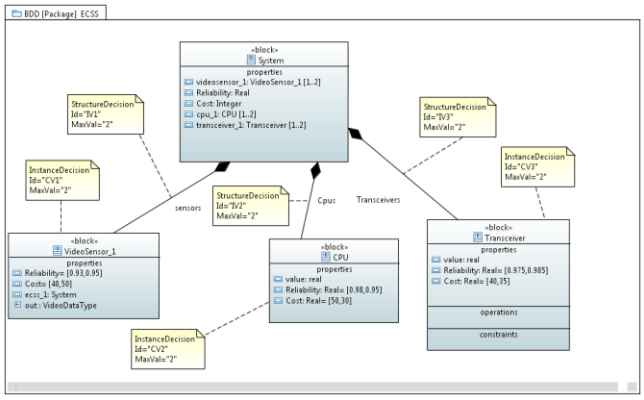

Figure 13 " BDD for UAV with decision points"

The reliability and the cost of each HW instance are integrated into the model of the system by associating a value list with blocks attributes. These values are presented in Table 5.

Table 5 "Reliability and cost for instances"

Component	Family	Mean Reliab.	Cost (€)
Sens. 1-3	OVH56	0.930-0.940	40-45
Sens. 4-6	OVH74	0.940-0.950	45-50
CPU 1-3	ARM7	0.950-0.970	30-40
CPU 4-6	ARM8	0.970-0.980	40-50
Trans. 1-3	TJA	0.975-0.980	35-37
Trans. 4-6	MCP	0.980-0.985	37-40

The MDO Context contains two objectives functions, the reliability R to be maximized and the cost C to be minimized. The objectives functions are inserted in the model with a constraint block. The MDO Context is represented by a parametric diagram, as shown in Figure 14 Parametric diagram for MDO Context."

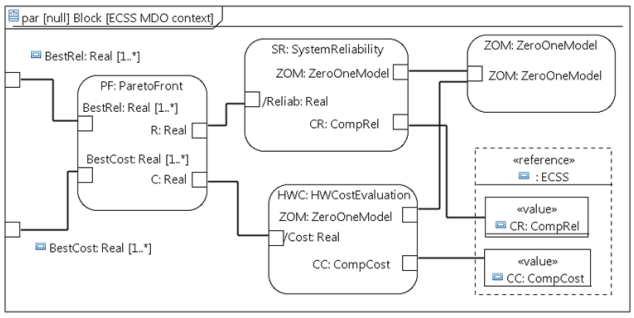

Figure 14 Parametric diagram for MDO Context

The objective functions have the following expression with the Boolean decisions variable of Equation 1:

$$\min C = \sum_{i,j,k} c_{jk}\left[x_{ijk} + exp\left(\theta_i \sum_{k} x_{ijk}\right)\right]$$

$$\max R = \prod_{k}\left[1 - \prod_{j,k}\left[1 - x_{ijk} r_{jk}\right]\right]$$

From a SysML/Papyrus model with six decision points, the three-steps algorithm described in paragraph 3.1 generates a 36-decision variables CSMOP problem. With the Labix CSP solver [25], a backtracking algorithm implemented in Python, and a posteriori objective function evaluation, we obtain 8,850 solutions in 11 minutes of computation time. The results are obtained with a JAVA implementation of algorithm 1 running on an Intel i5 3GHz machine with 4 GB RAM.

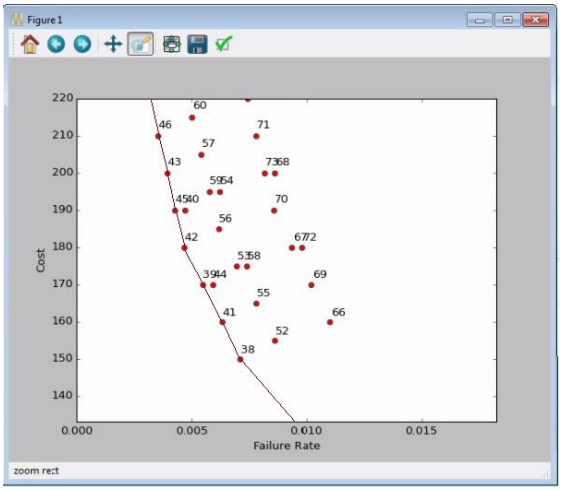

Figure 15 "Pareto frontier"

In Figure 15 the X-axis displays the Failure rate (1-Rs) instead of reliability Rs. Each point figures a solution to the CSMOP problem obtained with the Python Labix solver. The solid line figures the Pareto frontier, in a particular region of interest selected by the user from the set of solutions.

For a maximum cost of €190 and a failure rate < 100 10^{-6}, Table 6 presents the three best trade-off configurations selected by the designer.

Table 6 Cost and Failures rates

Sol.#	Sensor	CPU	Trans.	Cost	FR(10^{-6})
45	S_1+S_1	CPU_1	T_1+T_4	190	40
42	S_1+S_3	CPU_1	2xT_1	180	50
39	S_1+S_3	2xCPU_1	2xT_1	170	60

The optimal HW platform for reliability and cost consists of CPU_1 and a set of sensors and transceivers. CPU_1 is a multi-core platform: an Exynos 5422 Octa host processor, with four "Big" cores and four "Little" cores.

The HW components being chosen, the position of the camera in the UAV structure must be precisely determined. From a MBSE point of view, this problem deals with attributes values of the HW blocks. It corresponds to a value decision problem, where value types can be real (continuous problem). In this case, we can use "Values decision points" connected to class attributes and a solver such as PyOpt [10] for solving the CSMOP problem. The results are available in [29]. For the UAV system designer, next conflicting objectives to be optimized are the power dissipation and the computational capability. The allocation of software onto hardware helps to minimize these goals.

5.3 Software to Hardware allocation

The CPU_1 contains four powerful "Big" Cortex A15 cores and four slower "Little" Cortex A7 cores. The maximum dissipated power is 4 Watt for big cores and 1 Watt for little cores. Another feature of CPU_1 is that unused cores can be switched-off, rather than left idle. Thus, unused cores suffer no static leakage or dynamic switching power, and the minimum amount of used core shall be preferred to minimize dissipated power.

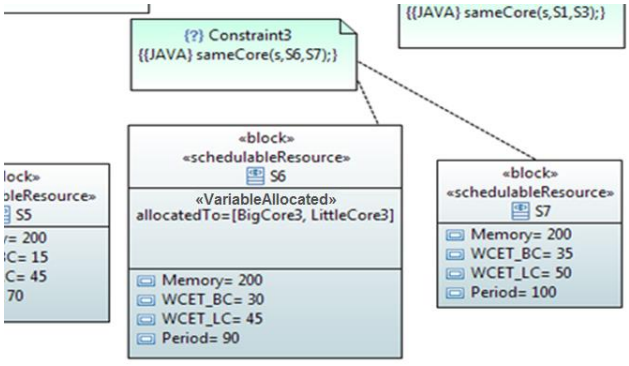

Figure 16 "BDD with variable allocation and global constraint"

The SW application of the UAV includes a set of scheduling resources for the mission realization (Path calculation) and a set for picture processing, obstacle detection and trajectory replanning.

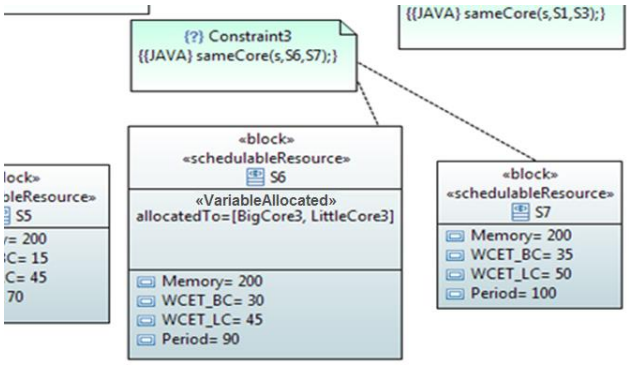

Figure 16 presents a BDD from the SW package, including the SW/HW allocation. The «*VariableAllocate*» stereotype defined in paragraph 3.1.2, Figure 6 is used for the variable allocation of "S_6" obstacle detection resource. For each resource, the "*WCET_BC*" and the "*WCET_LC*" attributes give the WCET value for one processor core. Other resources are characterized in Table 7. The "*SameCore*" constraint, equivalent to "*SameTarget*" constraint in Section 2, specifies that S_6 and S_7 resources shall be allocated to the same core, because they share a large amount of data for picture processing.

Table 7 Schedulable resources for UAV

Id	Name	WCET for BigCore(ms)	Period (ms)	Heap Memory size(KB)
S_0	Get Frame	10	80	300
S_1	Show Picture	20	90	100
S_2	Filtering 1	30	100	200
S_3	Filtering 2	30	100	100
….				
S_8	Obstacle detection	30	90	200
S_9	Trajectory Replanning	35	100	200

The SysML model includes also capacity constraints, presented in section 2, for each core. The "CoreMemory" constraint limits the core memory usage for the SW allocation and the utilization factor constraints limit the ratio between WCET and Period.

For a configuration "C_{1g}"with 10 resources allocated to 8 cores we obtain 3,135 solutions (Figure 17) with a resolution time of 0.5s. The the Eclipse/CHOCO environment was running on an Intel i5 3GHz machine with 4 GB RAM. For each set of distinct solutions, two objectives are calculated. First, the CPU dissipated power with a scalar product. Second, the number of allocated cores. These two objectives shall be minimized by the designer to obtain optimal solutions. For "C_{1g}" configuration we obtain 40 optimal solutions presented in Table 8 and Figure 17. A scheduling analysis if performed on each optimal solution (last column of Table 8).

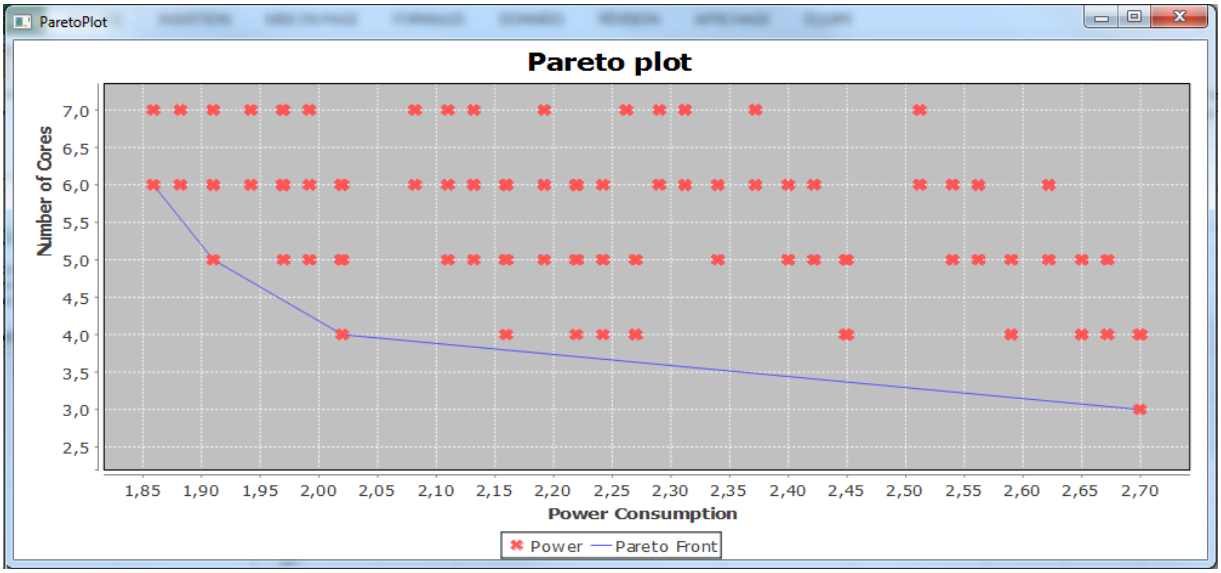

Figure 17 "Solutions for C_{1G}"

Table 8 Optimal solutions for C_{1G}

Core number	Possible allocation	Little Cores load (%)	Big cores load (%)	Dissipated Power (w)	Schedulable
3	1	0	1.799	2.699	yes
4	9	0.6	1.266	2.02	no
5	18	1.044	1.118	1.91	no
6	12	1.687	0.904	1.859	no

5.4 Papyrus tool and new plug-in

To support the methodology depicted by Figure 1, a plug-in has been added to the Papyrus modeling environment [22] with the Eclipse Modeling Framework (EMF) [21]. The plug-in uses open-source software. The three following features are supported by the plug-in and the profile presented in Figure 18:

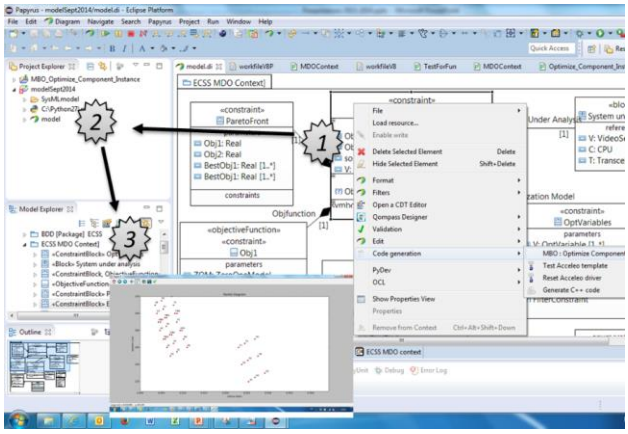

Figure 18 Papyrus tool and plug-in

1. The designer uses stereotypes to add decision points to his/her model. The Papyrus tool supports UML profiling by providing extensions to UML-based profiles. The stereotypes we propose are implemented in a custom profile. Therefore, the designer can define his/her own graphical or tabular notation to model the decision points described previously. In this step, he or she defines an optimization context diagram, including constraints blocks for objectives functions.
2. From the optimization context block, the designer generates a JAVA code similar to the one presented by listing 1. The source code is generated with a plug-in we have developed. The plug-in uses the XTEND language [30] for retrieving information from the model, with the template expressions feature. Template expressions (Listing 2) are multi-line strings within triple quotes and interpolated values from the model in French quotation marks. The result of this step is an updated JAVA file for the CSP problem, in the same Eclipse workspace but in a different project.

Listing 2 : Template expression with XTEND

```
«IF VarKind==Allocation»
# // Schedulable Resource number
private static final int S_MAX = «ResourceList.getNum()»
private static final int C_MAX = «CoreList.getNum()»
Solver solver = new  Solver("Allocation problem"); '''
«ENDIF»
```

3. Run the JAVA code generated in step 2, with a project using the selected solver. The results are displayed with a Pareto diagram. Other CSP solvers may be used at this step. In previous work [31], [29] we have experimented the Labix solver [31] for instance and structure decision points and the PyOpt solver for value decision points. Both solvers are written in Python and can be integrated in the Eclipse environment.

4. From the Pareto diagram, the table of optimal solutions can be displayed by the user. In this table, the detail of each solution is given, with the values of decision variables such as allocations or component choices. With this table, it is possible for the user to highlight the corresponding values in the model. This manual operation will be automated in the future with a new feature. This feature will allow round-trip optimization, by freezing decision points values after a first optimization step, and by adding other decision points in a second step. This iterative method reduces the complexity of the problem induced by the number of decision points.

6. Related Work

In [32], *Min et al* propose a multi-objective optimization from SysML model, by using the *ModelCenter* commercial tool. *ModelCenter* is a graphical environment for analysis and optimization. The designer provides a structural description of the system with SysML blocks that include properties. Then the block properties are connected one by one to a ModelCenter analysis block in a parametric diagram. This matches one of our decision points, the "Value decision", using connection with a commercial tool.

In [12] the authors propose a transformation from a feature model [17] to a mathematical representation of an optimization problem. Then a solver solves a combinatorial problem. Feature models represent all the products of a software product line. They are used in the whole product line development but not in MBSE with SysML.

With the COMPLEX methodology proposed in [33], the designer creates the various alternatives of allocations with the MARTE profile [34] and annotations. The alternative creation presented in Figure 19 is often manual: several variants can be missed and the size of the design space is a severe limitation. Also, in MARTE, the allocation semantics is ambiguous and unusable when numerous alternatives of allocations have to be modeled. For the transformation to analysis model and problem solving, the COMPLEX methodology proposes exhaustive search and the results are obtained by simulation. On a multi-core HW platform, the number of possible allocations can be significantly higher than on other HW platforms and the total simulation time becomes a problem.

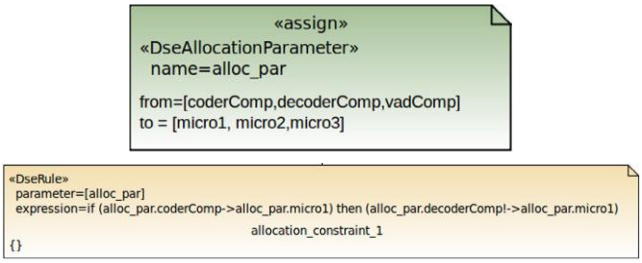

Figure 19 "Allocation with MARTE and COMPLEX"

7. Conclusions

The paper discusses trade-off analysis in a MBSE approach that associates SysML with so-called "decision points". Whether the approach is instantiated on SysML, the concept of decision points is not specific to one MBSE language and enables covering the four kinds of decisions problems largely found in system engineering: instance decision, redundancy, values and allocation problem.

For SysML, the paper proposes new stereotypes (decision points, global constraints and optimization context) extending the initial model without variants for trade-off analysis. Then, the algorithm proposed in the paper transforms the extended SysML model into a CSP optimization problem (*CSMOP*); the process includes decision variables, constraints and objectives functions. Several solvers have been benchmarked in order to address this *CSMOP* problem: CHOCO [9], PyOpt [10] and Labix [25]. The designer selects a solver, stored in the model, according to the kind of decision points, and according to the strategy required by the problem resolution. The methodology was tested on a multi-core UAVs model and validated with the plug-in that we have developed using Papyrus and Eclipse.

Unlike approaches published in [12] [32] [33], the one discussed in this paper allows to model the entire problem at the SysML model level, without programming code at the optimization tool level. Instead of manually connecting the SysML model with an optimization solver, it is possible to generate the problem description file using a SysML extension and our plug-in. This is particularly useful at early stage of design, when the exploration space is very large. With the proposed methodology, the design space exploration is more efficient, reducing the number of possible solutions before a detailed analysis such as scheduling analysis.

In the near future, the algorithm described in Section 4 will be optimized in terms of integration into the Papyrus tool, the purpose being to integrate and to compare several search strategies for the CSP problem resolution. The tool be improved with the highlighting of decision points values in the model, after a user selection of a particular optimal solution. This will allow iterative optimization by fixing decision points values after a first optimization and by adding new decision points. Another algorithm will be developed to help the user for the solver choice. Integration of detailed analysis, different from scheduling analysis for optimal solutions, will be studied too.

8. REFERENCES

[1] INCOSE, "Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities V3.1", Cecilia Haskins, CSEP, 2007.

[2] A. W. Wymore, "Model-based systems engineering", CRC Press, 1993.

[3] SysML, «"OMG Systems Modeling Language (OMG SysML™) V1.5",» 2017. [En ligne]. Available: http://www.omg.org/spec/SysML/1.5/.

[4] S. M. A. &. S. R. Friedenthal, "A practical guide to SysML : the system modeling language", Morgan Kaufman, 2014.

[5] M. Chern, «"On the computational complexity of reliability redundancy allocation in a series system,",» *Operations research letters*, vol. 11, n° %15, pp. 309-315, 1992.

[6] D. W. &. S. A. E. Coit, "Optimization approaches to the redundancy allocation problem for series-parallel systems," in Proceedings of Fourth Industrial Engineering Research Conference Proceedings, 1995, pp. 342-349.

[7] Ø. M.-P. B. O. J. O. G. K. &. S. A. Haugen, «"Adding standardized variability to domain specific languages," in Proceedings of : Software Product Line Conference,» 2008, p. 139–148.

[8] I. DAS, «"A preference ordering among various Pareto optimal alternatives, ",» *Structural optimization*, vol. 18, n° %11, pp. 30-35, 1999.

[9] CHOCO, ""Choco solver"," EMN, 2016. [Online]. Available: http://choco-solver.org/.. [Accessed 2016].

[10] R. E. J. P. W. &. M. J. R. Perez, «"pyOpt: a Python-based object-oriented framework for nonlinear constrained optimization. Structural and Multidisciplinary Optimization",» *Structural and Multidisciplinary Optimization*, vol. 45, n° %11, pp. 101-118, 2012.

[11] UML2, «"OMG. OMG Unified Modeling Language (OMG UML™) V2.5",» 2007. [En ligne]. Available: http://www.omg.org/spec/UML/2.5.

[12] P. &. K. H. D. Limbourg, «"Multi-objective optimization of generalized reliability design problems using feature models—A concept for early design stages",» *Reliability Engineering & System Safety*, vol. 93, n° %16, pp. 815-828, 2008.

[13] A. B. B. G. L. K. A. &. M. I. Aleti, «"Software architecture optimization methods: A systematic literature review,",» chez *IEEE Transactions on Software Engineering*, vol. 39, 2013, pp. 658-683.

[14] F. V. B. P. &. W. T. Rossi, Handbook of constraint programming, Elsevier, 2006.

[15] N. R. G. &. L. X. Jussien, «"Choco: an open source java constraint programming library," CPAIOR'08 Workshop on Open-Source Software for Integer and Contraint Programming (OSSICP'08),» 2008, pp. 1-10.

[16] M. V. G. J. &. B. J. Svahnberg, «"A taxonomy of variability realization techniques,",» *Software: Practice and Experience*, vol. 35, n° %18, p. 705–754, 2005.

[17] K. C. S. H. J. N. W. a. P. A. Kang, "Feature-Oriented Domain Analysis (Foda) Feasibility Study,", Software Engineering Institute, Carnegie Mellon University, 1990.

[18] I. &. F. K. Reinhartz-Berger, «"Comprehensibility of orthogonal variability modeling languages: the cases of CVL and OVM," in Proceedings of the 18th International Software Product Line Conference,» vol. 1, ACM, 2014, pp. 42-51.

[19] P. H. D. P. G. a. J. J. H. Feiler, "The architecture analysis & design language (AADL): An introduction.", Carnegie-Mellon Univ Pittsburgh PA Software Engineering Inst., 2006.

[20] B. G. S. Sélic, "Modeling and Analysis of Real-Time and Embedded Systems with UML and MARTE", Morgan Kaufmann, 2014.

[21] Eclipse, «"Eclipse Modeling Framework",» 2016. [En ligne]. Available: http://www.eclipse.org/modeling/emf/.

[22] Papyrus, «"Outil Papyrus",» CEA, 2015. [En ligne]. Available: https://eclipse.org/papyrus/.

[23] I. P. J. C. &. M. I. Gent, «"Minion: A fast scalable constraint solver" In ECAI,» 2006.

[24] L. &. B. F. Granvilliers, «"Algorithm 852: Realpaver: an interval solver using constraint satisfaction techniques",» *ACM Transactions on Mathematical Software (TOMS)*, vol. 32, n° %11, pp. p. 138-156., 2006.

[25] G. Niemeyer, 2016. [En ligne]. Available: https://labix.org/python-constraint.

[26] K. R. &. W. M. Apt, "logic programming using ECLiPSe,", Cambridge University Press, 2006.

[27] J. P. F. C. K. K. C. &. B. L. F. How, «"Increasing autonomy of UAVs",» *Robotics &Automation Magazine*, vol. 16, n° %12, pp. 43-51, 2009.

[28] D. &. D. U. Weatherington, «"Unmanned aircraft systems roadmap, 2005-2030",» Deputy, UAV Planning Task Force, OUSD (AT&L), 2005.

[29] P. S.-S. P. D. H. Leserf, «"Multi-Domain optimization with SysML modeling," in Proceedings of ETFA 20th IEEE conference,» Luxembourg, IEEE, Sept. 2015, pp. 1-8.

[30] L. Bettini, "Implementing Domain-Specific Languages with Xtext and Xtend,", Packt Publishing Ltd, 2013.

[31] P. S.-S. P. D. H. J. &. C. K. Leserf, «"Architecture Optimization with SysML Modeling A Case Study Using Variability",» chez *MDE and software developement - CCIS*, vol. 580, Springer, Déc. 2015.

[32] B. I. K. A. A. &. P. C. J. Min, «"Process integration and design optimization for model-based systems engineering with SysML," In ASME 2011 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference,» American Society of Mechanical Engineers, 2011, pp. 1361-1369.

[33] F. e. a. Herrera, «The COMPLEX methodology for UML/MARTE Modeling and design space exploration of embedded systems,» *Journal of Systems Architecture*, pp. 55-78, 2014.

[34] MARTE, «"UML Profile for MARTE: Modeling and Analysis of Real-Time Embedded Systems V1.1",» 2011. [En ligne]. Available: http://www.omg.org/spec/MARTE/1.1.