|  |  |  |  |
| --- | --- | --- | --- |
| **Livrable** | | | |
| **Tâche 1** |  | | **Date : 04/04/2025** |
| L1.1.0 Définition et caractérisation multicritères applications benchmarks robotique mobile autonome avec CONOPS | | | |
| **Responsable** | | ENSTA | |
| **Participants** | | ENSTA | |

|  |  |  |  |
| --- | --- | --- | --- |
| Contributeurs : | Nom | Organisation | Rôle |
| Responsable du livrable | Thomas Rigaut | ENSTA | Ingénieur de recherche |
| Responsable scientifique | Omar Hammami | ENSTA | Professeur |
| Co-auteurs | Thomas Rigaut | ENSTA | Ingénieur de recherche |
| Thomas Vita | ENSTA | Ingénieur stagiaire |
|  |  |  |
| Resp. revue interne | Omar Hammami | ENSTA |  |
|  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Historique du document: | | | | |
| Version | Date | Modifications | Etat[[1]](#footnote-2) | Distribution |
| V0.1 | 04/04/2025 |  | Ebauche |  |
| V1 | 11/04/2025 |  |  |  |
| V2 | 17/04/2025 |  |  |  |

Abstract

The Model Architect is the actor of the design process of complex systems who assesses the system architectures defined by the System Architect, using simulation. His aim is to build the appropriate model regarding the constraints of the simulation demand.

In this deliverable, the Model Architect Problem (MAP) is defined as scheduling Model Architect tasks in order to answer the simulation demand. It uses a multi-criteria scheduling description using a three fields notation (resources, constraints, objectives). Several variants of the MAP are proposed.

However, the definition of this scheduling problem introduces new concepts (like simulation quality, the notions of risk, uncertainty, utility), model quality and model composability are defined and links with the MAP are specified.

Several mechanisms used to partially automate the Modelling and Simulation process will be introduced in this deliverable.

Finally, the study carried out on a Heterogeneous Robotic Swarms case study concerning the optimization of the on-board electric network of a mild hybrid UAV and Terrestrial Robot is described.

Table des matières

[Abstract 2](#_Toc194743023)

[Table des matières 3](#_Toc194743024)

[Liste des figures 4](#_Toc194743025)

[Liste des tables 4](#_Toc194743026)

[Liste des acronymes et abréviations utilisés 5](#_Toc194743027)

[1 Introduction 6](#_Toc194743028)

[2 Multi-criteria scheduling and the Model Architect Problem (MAP) 6](#_Toc194743029)

[2.1 Introduction to multi-criteria scheduling 7](#_Toc194743030)

[2.2 Formulation of the Model Architect Problem (MAP) 8](#_Toc194743031)

[2.2.1 Model Architect Problem 8](#_Toc194743032)

[2.2.2 Model Architect Problem 9](#_Toc194743033)

[2.3 Conclusion 9](#_Toc194743034)

[3 Model quality 10](#_Toc194743035)

[3.1 Introduction 10](#_Toc194743036)

[3.2 State of the art 10](#_Toc194743037)

[3.2.1 Risk 10](#_Toc194743038)

[3.2.2 Uncertainty 10](#_Toc194743039)

[3.2.3 Utility (decision making) 11](#_Toc194743040)

[3.2.4 Model quality in M&S 12](#_Toc194743041)

[3.2.5 Composability 13](#_Toc194743042)

[3.2.5.1 Syntactic composability 13](#_Toc194743043)

[3.2.5.2 Semantic composability 13](#_Toc194743044)

[3.3 Model quality in MAP 14](#_Toc194743045)

[3.3.1 Model semantics definition 14](#_Toc194743046)

[3.3.2 Sub-problem 1: model specification for composability 15](#_Toc194743047)

[3.3.3 Sub-problem 2: integration of models for composability 15](#_Toc194743048)

[3.3.4 Other sub-problems 15](#_Toc194743049)

[3.4 Conclusion 15](#_Toc194743050)

[4 Automation methods for numerical simulation 15](#_Toc194743051)

[4.1 Introduction 15](#_Toc194743052)

[4.2 MIC definition and simulation automation with pseudo-MICs 16](#_Toc194743053)

[4.2.1 MIC definition 16](#_Toc194743054)

[4.2.2 Simulation automation with pseudo-MICs 16](#_Toc194743055)

[4.3 Automation mechanisms 18](#_Toc194743056)

[4.3.1 MIC2MO 18](#_Toc194743057)

[4.3.2 MIC2V 19](#_Toc194743058)

[4.3.3 CompMICs 19](#_Toc194743059)

[4.3.4 MO2MIC 20](#_Toc194743060)

[4.4 Implementation 20](#_Toc194743061)

[4.5 Conclusion 21](#_Toc194743062)

[5 Case study: Heterogeneous Robotic Swarms 21](#_Toc194743063)

[5.1 Introduction 21](#_Toc194743064)

[5.2 Conclusion 23](#_Toc194743065)

[6 Conclusion 23](#_Toc194743066)

[Références / Bibliographie 23](#_Toc194743067)

[Annexes 26](#_Toc194743068)

[Annex 1 – Benchmark modeling with BPMN 27](#_Toc194743069)

[Annex 2 – Pragmadev 29](#_Toc194743070)

[Annex 3 – LTL – Linear Temporal Logic – Model Checking 31](#_Toc194743071)

[Annex 4 – UAV – Tello - Husky 34](#_Toc194743072)

[Annex 5 – NAF v.3 – MEGA HOPEX v.5 – BPMN – HRS Modeling and Simulation 35](#_Toc194743073)

[Annex 7 – FACE 36](#_Toc194743074)

Liste des figures

[Figure 1: MIC classes 16](#_Toc452473468)

[Figure 2: Example of a MIC2MO transformation of the pseudo-MIC of a lithium battery 19](#_Toc452473469)

Liste des tables

**Aucune entrée de table d'illustration n'a été trouvée.**

Liste des acronymes et abréviations utilisés

| Abbreviation | Definition |
| --- | --- |
| CBSE | Component based systems engineering |
| CCO | Cost-Constrained Optimization |
| CSE | Computational Science and Engineering |
| DM | Decision Maker |
| MA | Model Architect |
| MAP | Model Architect Problem |
| MAUT | Multi-Attribute Utility Theory |
| MCDA | Multi-Criteria Decision Aid |
| MDCM | Multi-Criteria Decision Making |
| M&S | Modelling and Simulation |
| RCO | Reliability-Constrained Optimization |
| RD | Robust Design |
| RID | Risk Informed Design |
| SA | Simulation Architect |
| SIM | Simulation et Ingénierie Multidisciplinaire |
| V&V | Validation & Verification |

# Introduction

The Model Architect is a new role that has been defined within the SIM project. The Model Architect is a stakeholder of the conception cycle of complex systems who configures the architecture of models, builds the workflow of generation and assembly of models, specifies models for the model providers, integrates them in the simulation and then simulates the system of interest.

In the first section of this document, the Model Architect Problem (MAP) is introduced and defined in terms of multi-criteria scheduling. This problem formalizes the workflow of tasks performed by this stakeholder when a multi-physics simulation demand is performed. A trade-off between the quality of the simulation, the resources used and the time required to get the simulation results is proposed because those criteria can be conflictual in some cases. That induces the creation of a Pareto Front.

Then, the next section concerns the notion of model quality, which must be defined and formalized in order to solve the Model Architect Problem. A state of the art is initially presented, and the links with the notions of risk and uncertainty are described. The notion of utility is also introduced because the aim of the simulations performed is to make decisions. That is why the decision-making purpose and the notions induced like utility are dealt with. The concept of composability of simulation models is also introduced because the simulation of complex models composed of several sub-models is considered. Then, those notions are used to reinterpret the MAP with model quality explained.

The next section will introduce and develop four mechanisms to partially automate the Modelling and Simulation process.

The last section exposes the work done on the Renault case study (on-board electrical network of a mild hybrid vehicle) whose description can be found in deliverable L4.2. Several simulation models for the same components are developed to treat the composability problem, and an alternative formulation of the optimization problem is proposed.

# Multi-criteria scheduling and the Model Architect Problem (MAP)

The Model Architect is the actor who has to assess the system architecture defined by the System Architect. And for that purpose, he uses simulation. In fact, his aim is to build the appropriate model regarding the constraints of the simulation demand.

The Model Architect Problem consists in scheduling the whole workflow of the simulation process of a complex system, from the simulation demand by the System Architect (given an architecture of the system) to the return of the simulation result. This includes collecting the models of the sub-systems, assembling them and then simulating them. However, several conflicting criteria must be optimized: the time of the process, the resources used and the quality of the simulation.

The first subsection will introduce multi-criteria scheduling theory, its aims and a notation that is commonly used in the literature to characterize classes of scheduling problems. The second subsection deals with several multi-criteria scheduling formulations of the Model Architect Problem.

## Introduction to multi-criteria scheduling

According to (Pinedo, 1995), "scheduling concerns the allocation of limited resources to tasks over time. It is a decision-making process that has as a goal the optimization of one or more objectives". Hence, because scheduling concerns generic terms such as tasks and allocation, this domain is effectively used in various fields of application, from computer operating systems to project management.

In order to define a scheduling problem, we are given:

- jobs ( is the job number );

- is composed of operations ( is the operation of );

- machines ( is machine ). The machines are also called the resources.

When solving a scheduling problem, the aim is to assign the jobs to the machines while respecting some constraints and optimizing given criteria. In order to describe the configuration, constraints and objectives of a scheduling problem, a notation consisting of three fields , has been developed in the literature. Each field is a list of words separated with commas. Hence, a specific characterizes a class of problems. The meaning of each field is the following:

- defines the environment in terms of resources (machines);

- lists the constraints of the problem;

- specifies the optimization criteria.

In order to ease the comprehension, we present in the following a simple scheduling problem:

The first field means that there is only one machine in the environment. The second field means that there exists a precedence relation between the jobs that will be performed. And finally, the objective is described by . In a schedule, the completion time of job is . And is defined by . It is the maximal completion time over all jobs for a schedule. The objective in is to minimize .

The three fields notation allows to differentiate between different classes of problems. It groups problems from different domains of application, using their commonalities from a scheduling viewpoint. Thereby, this notation allows to search in the literature for a solution or a property (for example, the computational complexity) of a problem using its class.

Then, can represent a single criterion as well as multiple criteria. The latter case corresponds to multi-criteria scheduling which is a branch of scheduling theory that deals with multi-objective problems. Hence, the aim is to optimize several conflicting criteria, and to provide Pareto optimal schedules. To be more understandable, concepts concerning multi-objective optimization must be defined. First, a multi-objective optimization problem is an optimization problem with multiple objective functions. Let be the feasible set of decision vectors, and the criteria space of finite dimension. Therefore, . The multi-objective optimization problem is defined by:

With this formulation, is the objective function to be minimized, and the functions define the constraints of the problem. The term means that . Then, optimality in the multi-dimensional criteria space must be defined. A partial preorder can be defined in the criteria space. , we have:

The definition of optimality that is used is the one of Pareto optimality. A list of definitions concerning this notion is therefore provided:

Definition of a weak Pareto optimum:

The set of weak Pareto optima is noted .

Definition of a strict Pareto optimum:

The set of strict Pareto optima is noted . We have .

Optimality in our optimization problem is defined in the sense of Pareto optimality. A Pareto optimal solution is a set of elements whose any objective cannot be improved without degrading another objective. The goal is to approximate the set of Pareto optimal solutions which is also called the Pareto front.

The final notion to be introduced in this section concerns robustness considerations in scheduling theory. In fact, real-world problems are characterized by uncertainties on the parameters (for example, processing times). Therefore, a schedule that is valid and optimized in theory can become useless and infeasible in practice. In order to overcome this problem, a trade-off between the optimality and the robustness of the schedule regarding perturbations must be established. Two approaches to solve that are described in (T'kindt, 2006):

- the proactive approach. Some knowledge on the uncertainties of the problem is taken into account in order to search more robust solutions.

- the reactive approach. The schedule is modified in real time as unexpected perturbations arise.

In practice, a mean to add robustness to a schedule is to use the notion of flexibility. It eases the modification of a schedule. Three types of flexibilities exist according to (T'kindt, 2006): the sequencing flexibility, the temporal flexibility and the assignment flexibility.

Those concepts relating to multi criteria scheduling and robustness will be used to define the Model Architect Problem in the next section.

## Formulation of the Model Architect Problem (MAP)

In this section, two formulations of Model Architect Problem using multi criteria scheduling are proposed. They differ in the way they consider the availability of sub models.

### Model Architect Problem

The first definition of the MAP assumes that the whole set of models can be found in a “Model Factory”. Then, the Model Architect Problem can be expressed with the following form:

We have the resources . This means that we are given unrelated parallel machines. The machines are computers, engineers and models. Thus, they are heterogeneous.

The constraints of the problem are . The first one, , means that there are precedence constraints (between the tasks) forming a series-parallel graph. The second one, , denotes that machines (resources) can be unavailable during periods that are known in advance.

Finally, the optimization criteria are. This means that the goal is to reach the Pareto front of the following optimization criteria:

- which is the total time of the whole simulation process;

- which is the quality of the simulation;

- which represents the resources effectively used during the schedule.

### Model Architect Problem

For this second definition, let’s assume that some of the models are unavailable. Thus, the design and the collection of those models belong to the set of tasks to be scheduled. Compared to the previous definition of the problem, the models are removed from the resources . The difference between those two definitions is how the models are considered (resources or not). Moreover, the design and collection of those unavailable models belongs to the set of tasks to be scheduled. Hence, using the notation, the problem is still the same (). The difference lies in the fact that the models are not considered as being part of the resources .

## Conclusion

Classes of scheduling problems can be identified using the three fields notation:

The field defines the resources, the constraints and the optimization criteria of the given problem. These fields can be used to search in the literature for the complexity of the problem and a possible algorithm to solve it.

This notation was used to express the Model Architect Problem in terms of multi criteria scheduling. However, different definitions can be used, depending on the viewpoint concerning the relationships between resources and tasks.

Moreover, other definitions of the Model Architect Problem have to be defined in order to introduce the notions of robustness and flexibility in the evaluation of schedules. That is why the fields of multi-criteria fuzzy scheduling and stochastic scheduling must be investigated because they could possibly tackle those notions of robustness and flexibility.

# Model quality

## Introduction

In simulation-based engineering, simulation is used for a decision-making purpose, in order to choose between alternative architectures for example. The notions of risk and uncertainty have an importance in the decision-making field. Then, the concept of utility that is used to model the preferences of the decision maker over outcomes of interests is developed. In the first subsection, a state of the art concerning these terms is made, and the links with Modeling and Simulation (M&S) found in the literature are shown. Then, model quality assessment in M&S will be studied. Finally, a literature review of the composability of simulation models will be presented. Note that neither Renault nor Airbus Group provided internal references or definitions of the terms risk, uncertainty, utility and quality. The second subsection concerns model quality in the MAP (Model Architect Problem) because in the first section about scheduling, the quality was mentioned but not defined in the MAP.

## State of the art

This section introduces several concepts linked with the notion of simulation model quality in M&S: risk, uncertainty, utility and composability. Their connections through model quality are then presented. Finally, the notion of composability of simulation models is defined.

### Risk

According to the ISO 31000 standard, the risk is “the effect of uncertainty on objectives”. (Chaves, 2014) lists some of the definitions of risk used in the engineering field. The risk can be defined as:

is in fact a random variable that expresses the severity of the outcomes of an event (it could be a cost, for example), is the probability density function of , and are the limits for the severities of the outcomes. Hence, is the expected value of the risk (the first equation for the continuous case, the second for the discrete one). This definition takes into account the classical understanding of the risk of an event , which is:

Thanks to this last equation, we can deduce that the risk of an event is the severity of the risk outcome multiplied by the probability that the event will occur. This definition of risk can be found in (Van Bossuyt, 2013).

As it can be noticed thanks to the ISO definition of risk, the notion of risk is strongly linked with the one of uncertainty. The latter will be defined in the next subsection.

### Uncertainty

Uncertainty in the context of Modeling and Simulation (M&S) has been defined in (Walker, 2003) and (Oberkampf, 2002). According to (Walker, 2003), uncertainty is any departure from the unachievable ideal of complete determinism. Moreover, three dimensions of uncertainty in a model appear:

- The location of uncertainty: it describes where uncertainty arises in the model (context, model uncertainty, inputs, parameter uncertainty, model outcome uncertainty);

- The level of uncertainty: from determinism to indeterminacy;

- The nature of uncertainty: epistemic or variability uncertainty;

In (Oberkampf, 2002), a distinction is made between uncertainty and error, even if the notions of epistemic and aleatory (same as variability) uncertainty are used. Aleatory uncertainty is the inherent variation associated with the physical system or environment under consideration. Epistemic uncertainty is a potential inaccuracy in any phase or activity of the modeling process that is due to lack of knowledge. And error is a recognizable inaccuracy in any phase or activity of modeling and simulation that is not due to lack of knowledge.

Then, (Oberkampf, 2002) also defines the phases of modeling and simulation. They are, in chronological order:

- Conceptual modeling of the physical system;

- Mathematical modeling of the conceptual model;

- Discretization and algorithm selection for the mathematical model;

- Computer programming of the discrete model;

- Numerical solution of the computer program model;

- Representation of the numerical solution.

However, feedback can also propagate between these phases. (Oberkampf, 2002) applied the methodology to a missile flight example, listing the uncertainties and errors occurring during each M&S phase.

Another important field of literature about uncertainty concerns uncertainty propagation in simulation models. In fact, given the probability distributions of the random variables of a system (the parameters of the model, for example), the problem is to determine the probability distributions of the system output (the simulation outcomes). In (Chen S. H., 2007), five categories of uncertainty propagation methods for functions are listed:

- Simulation based methods;

- Local expansion based methods;

- Most probable point based methods;

- Functional expansion based methods;

- Numerical integration based methods.

The concept of uncertainty in M&S influences the decision making process and its associated risk. That is why it is strongly linked with the concept of utility that will be developed in the next section.

### Utility (decision making)

In decision-making, the aim of the decision maker (DM) is to choose between different alternatives . Each alternative can lead to a consequence . The utility is a function that represents the preferences of the DM over the consequences in order to discriminate alternatives. Hence, using the notion of utility as a criteria, the DM transforms his decision-making problem into an optimization problem. To make a decision, the DM has to choose the alternative which maximizes the expected value of the utility.

As the utility represents the preferences of the DM over outcomes, it also expresses the DM's behavior towards risk. Therefore, utility functions can express risk averse, risk seeking and risk neutral attitudes.

The theory of utility was developed by Von Neumann and Morgenstern in (Von Neumann, 1944). The Multi attribute utility theory (MAUT) was then created to consider multiple attributes in the utility function. In MAUT, the utility function takes values in a multi-dimensional space in order to manage trade-offs between different criteria.

Decision making in the field of M&S is an issue that has already been identified in the literature. For example, (Radhakrishnan, 2005) uses the concept of utility to solve the decision problem of model selection. The method developed, which is the model sufficiency estimation technique, is used to select the most useful model from a set of models, via the estimation of model utilities. An uncertainty analysis is then performed to determine the decision's confidence. The main interest of this article is that it considers model selection as a decision problem. However, this approach implies the availability of each model. A reverse uncertainty propagation method is also proposed. It deduces sufficiency requirements about model accuracy and cost, for example, from backward propagation of decision uncertainty. This mechanism determines how truthful a model needs to be in order to make decisions.

### Model quality in M&S

Model quality is quite developed and formalized in the field of software engineering. For example, (Crnkovic, 2004) classifies quality attributes for component-based systems. Attributes can be directly composable, architecture related, derived, usage-dependent or system environment context determined. This paper treats attributes such as reliability, availability, safety, confidentiality, integrity and maintainability.

On the contrary, the notion of model quality remains quite ambiguous in Modeling and Simulation. Many indicators may have an influence on a M&S application quality. Accuracy is not the only one. (Balci, 2004) develops a quality-centered approach for M&S applications (acceptability criteria, Verification and Validation (V&V) are dealt with).

However, attempts to treat model quality more formally have been made. The report from the Fidelity Implementation Group (Gross, 1999) describes the notion of fidelity in various activities of M&S. The concept of fidelity is relevant because it is what differentiates computer simulations from other computer programs. This report also provides a glossary of M&S terms and a mathematical formalism for fidelity standards.

Nevertheless, other concepts that can be partially mapped to quality have been developed in the literature. For example, the NASA introduces a credibility assessment scale for M&S results in its 7009 technical standard (Standard for models and simulations (NASA, 2008)). The credibility assessment scale is divided into 8 factors classified into 3 categories:

- the M&S development category (verification evidence factor, validation evidence factor);

- the M&S operations category (input pedigree evidence factor, results uncertainty evidence factor, results robustness evidence factor);

- the supporting evidence category (use history factor, M&S management factor, people qualifications factor).

Each factor is evaluated on a 5-levels scale. A global credibility factor is then deduced from the 8 factors. The two first categories are related to the intuitive notion of quality, mentioning verification and validation (V&V), uncertainties and robustness. Third category seems more subjective but has an influence on the confidence about simulation results.

### Composability

According to (Petty, 2003) and (Petty, A formal basis for a theory of semantic composability, 2003), composability is the capability to select and assemble simulation components in various combinations into valid simulation systems to satisfy specific user requirements. And in practice, they define two types of composabilities:

- the syntactic composability;

- the semantic composability.

These two concepts will be presented in the next subsections. In the frame of composability in Petty's work, a model M is defined as a computable function:

In this definition, is a non-empty set of states, is a set of inputs, is a set of outputs, and are vectors of integers. Hence, a simulation is the sequential execution of a model and is represented by a deterministic labeled transition system where is a model, and is the state model of . A trajectory in is a sequence of alternating states and inputs beginning with and defined by:

Therefore, there are terminating and non-terminating trajectories.

Then, in order to define composability, several concepts related to validity need to be defined. In M&S, the model being built refers to a natural system. A perfect model can be linked to that natural system. Hence, the validity of other models for the natural system is measured with respect to the perfect model. In order to assess the validity of the model, strong and weak bisimulations can be used to compare the executions of the two models. Bisimulations are binary relations between transition systems. Two systems are said to be bisimilar if they have the same behavior (if their moves can be mapped between each other). Moreover, equivalence relations and metrics can be employed to assess the validity.

Theoretical results exist in the literature that tackle computational problems in composability. For example, (Petty M. D., 2003) treats the computational complexity of the component selection problem for composition. This problem had already been identified and partially solved by (Page, 1999). In fact, the general component selection problem is NP-complete, even if the properties of a component or a selection are already known (via the use of an oracle). It allows to determine the intrinsic difficulty of the component selection without considering the difficulty of objectives decidability. Hence, other problems related to composability also need to be treated (from a computational complexity viewpoint): the determination of component and composition objectives, the validity determination and the composition complexity.

#### Syntactic composability

Syntactic composability refers to the property that simulation models can be connected in a proper manner so that they can exchange information during a simulation. According to Petty, the question is whether the components can be properly connected.

#### Semantic composability

Contrary to syntactic composability, the concept of semantic composability corresponds to whether simulation components can be meaningfully composed, that is to say to which extent a set of individually meaningful (validated) components can be composed into a meaningful (valid) global simulation model.

## Model quality in MAP

The Model Architect Problem (MAP) can now be reinterpreted as the state of the art for the concept of quality was defined in the last section. In fact, enhancing the quality of the simulation would lead to increased completion time to create the simulation models and then perform the simulation, but also to an increased need of resources . That is why the trade-off between used in the definition of the Model Architect Problem seems relevant. To the best of our knowledge, there is no tool that estimates this trade-off between in multi-physics simulations. There is even no solution in existing software to guarantee the quality of a simulation model.

### Model semantics definition

The concept of Model Identity Card (MIC) was proposed and is used in the SIM project to specify simulation models and then to verify that a simulation model fulfils the specification. Quality attributes are contained in the MIC description, because they translate a part of the model properties needed in order to run the appropriate simulation.

A first category of properties described in the MIC concerns the ports and variables of the considered model. They are characterized in order to guarantee the syntactic composability between the models. Indeed, these interfaces are the means by which models will exchange information during the simulation. Then, characterizing ports in the MIC would also characterize the syntactic composability.

Then, a second category of information named model quality can be found in the MIC. This category contains indicators from the credibility assessment scale of the NASA mentioned above:

- Validation of models;

- Verification of models;

- Input pedigree;

- Uncertainties;

- Robustness;

- History.

Those indicators are evaluated on a 5 levels scale like the NASA sub factors. However, those levels only indicate the advancement in the associated processes. They do not carry information about the objectives of those processes and the properties they establish. Hence, in order to guarantee the semantic composability of simulation models, the existing view concerning model quality in the MIC is not sufficient.

Semantic characteristics of simulation models need to be defined to characterize the behavior and meaning of those models. Different levels of semantics can be defined by abstraction and refinement processes because concrete semantics are not always suitable or available to describe the expected properties of the model. This mechanism is exploited in the field of abstract interpretation (Cousot, 1992) in computer science. This theory of sound approximation of programs semantics may describe the behavior models. The problem is to find a suitable level of abstraction to describe the behavior of the model for validation purposes. Those semantical aspects are needed to treat the model selection problem because the current definition of the MIC (which mainly relies on syntactic properties of models) does not sufficiently addresses the behavior of the model.

### Sub-problem 1: model specification for composability

The first problem concerns the specification of the simulation models to be composed. When the MA is given an architecture of the system to be simulated and a simulation demand, and after decomposing topologically the global simulation model into a network of sub models, the problem is priori how to express and specify the quality properties that would be implemented by the global model and the sub models, in order to guarantee the syntactic and semantic composabilities when the models would be integrated and the simulation performed.

### Sub-problem 2: integration of models for composability

The second problem concerns the posteriori check of the quality properties (specified in the first problem) during the integration of the sub models, at both sub model and model levels, before running the simulation. The aim is to check the composability, specifically the semantic one.

### Other sub-problems

Another problem to be considered is the one of model selection. Considering a demand for a network of models via the specifications provided by the first problem, and a set of eligible models for each sub model of the network, how to select a candidate for each model of the network and create it effectively in order to maximize the degree to which the quality metrics are implemented.

One last problem is the choice of an appropriate abstraction for the semantics of a model which allows the verification of the specification of the model without being too abstract.

## Conclusion

This section focused on the quality issues that can be found in the MAP. Several problems concerning quality that are necessary to solve the MAP were identified, concerning both the definition of the specification of a model and its verification.

# Automation methods for numerical simulation

## Introduction

The MIC (Model Identity Card) is a formalism which purpose is to allow the specification and the characterization of simulation models. A MIC can be viewed as a demand of a simulation model. This demand is made by an actor called the Model Architect (MA). The role of this actor is to perform the simulation of a system when given a simulation demand of the system of interest by the system architect (SA). For that purpose, the MA specifies the MICs of the simulation models to be produced. Those MICs serve as an input for the third type of actor, the Model Providers (MP). They have to produce and furnish the simulation models corresponding to the specified MICs. Finally, the MA integrates those models in a network of models according to its simulation architecture, in order to run the simulation and interpret its results.

This process means that MICs contain necessary information for the fulfillment of the M&S (Modelling and Simulation) process. Since the MIC is defined by a formalism, it can be interpreted. Hence, several tasks of the M&S process can be automated using the MIC as an input. In this document, four mechanisms dealing such tasks are introduced and developed. They are:

- MIC2MO: it generates the backbone of a simulation model from its MIC;

- MIC2V: it creates a program that automatically checks requirements on simulation traces;

- CompMICs: it compares two MICs and gives a measure of similarity between those two MICs;

- MO2MIC: it creates the MIC of a model directly from the source code of the model.

Those mechanisms were developed within the scope of Modelica simulation models. The availability of tools dealing with such models (OpenModelica and Python to treat the results of the simulations), combined with the easy access to the Modelica specification (in order to get the BNF grammar of the language) influenced that choice. However, the application to Modelica models is a proof of concept. Those mechanisms could be extended to other simulation languages provided that their BNF grammar is available.

Finally, those four mechanisms, when combined, allow to generate a network of models, to populate it and to check requirements on its simulation.

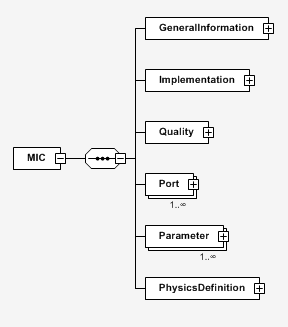
## MIC definition and simulation automation with pseudo-MICs

### MIC definition

The MIC is defined as a common vocabulary to specify and characterize simulation models. It characterizes a model through the interfaces definitions.

A MIC is composed of different classes of information. They are: General information, implementation, quality, Ports, Parameters, Physics definition.

Figure 1: MIC classes



Each class contains fields of information.

### Simulation automation with pseudo-MICs

The work presented in this document deals with the partial automation of the activities linked with the model associated to a MIC. A partial automation would ease the run of simulations in order to treat the quality aspects of the simulation request.

In this work, the only information that is kept from the MICs is the one that is relevant in an automation perspective. This information is the one contained in the classes quality, ports and parameters.

However, with an automation purpose, this information is not sufficient to generate backbones of networks of simulation models and to compare simulation models via their specifications. In the former case, the information concerning the architecture of the simulation lacks, because it allows to describe networks of connected models. In the latter case, the description of the internal components of the model lacks. For example, it is not possible to discriminate between the MIC of a resistor and the MIC of a Lithium battery when the comparison is only based on the definition of the interfaces of the models. However, those elements differ by their parameters and internal variables.

For those reasons, in this work, the specifications of simulation models are not MICs but variations from MICs that are in fact pseudo-MICs. With those pseudo-MICs, a simulation model is specified by the definition of its constituting elements (sub-models, ports, variables, parameters). A hierarchical representation of those elements in the form of a tree is used.

A pseudo-MIC can either characterize a simulation model, a port, a variable or a parameter. A model can contain models, ports, variables and parameters. A port can contain variables.

However, this hierarchical representation of the specification of the model is not sufficient. Adding information about each constituting element of the model would induce more detailed specifications to allow to differentiate them. The problem is to define the granularity of this information. The pseudo-MIC of a model must remain simple to define while it has to embed enough information to characterize the elements. Some of this information is already defined in MICs and will be used in the present work.

For that purpose, properties can be linked to each constituting element, according to its type.

For a model, the allowed properties are:

- The connection: At the level of model , it specifies a connection between two ports and as . However, if belongs to model and belongs to model , and must be sub-models of . Moreover, and must be consistent with each other, that is to say that they must contain the same number of variables with the same names and units. This latter condition ensures the syntactic composability, that is to say that the two ports can be effectively connected.

- The timestep: a fixed timestep and its numerical value can be specified, or a variable timestep also. Only one timestep can be defined per model.

- The timestep tolerance: a range specifies the allowed timesteps.

- The propositional variable: a propositional variable of model is the comparison of a variable that is hierarchically a descendant of with a floating value. The comparison operators that can be used are . An example of propositional variable is: .

- The requirement: a requirement of model is defined by a MITL (Metric Interval Temporal Logic) formula on the propositional variables of . The grammar used for MITL formulas is the following one:

Where is a propositional variable, and are formulas, and . is the until operator. Other operators of logics can be defined:

Other temporal operators can be derived:

Where means True, and False. means Future and Globally.

For a variable, the allowed properties are:

- Min: it specifies a minimum value for the variable;

- Max: it specifies a maximum value for the variable;

- Flow: if this property is defined, then the variable is a flow variable;

- Init: this specifies the initial value of the variable;

- Unit: it characterizes the unit of the variable.

For a parameter, the allowed properties are:

- Min: it specifies a minimum value for the parameter;

- Max: it specifies a maximum value for the parameter;

- Value: this specifies the value of the parameter;

- Unit: it characterizes the unit of the parameter.

The recursive representation of a model from the description of its constituting elements associated to the definition of the properties of these elements is the pseudo-MIC representation that will be used in the work described in this document.

## Automation mechanisms

The pseudo-MIC described in the previous section is a formalism that can be interpreted. This allows, given the information embedded in MICs, the automation of several mechanisms of the M&S process. Four such mechanisms are presented in the following subsections:

- MIC2MO: its transforms a pseudo-MIC into the backbone of a simulation model;

- MIC2V: it generates a program that will check the requirements defined in the pseudo-MIC of a model along the simulation traces;

- CompMICs: it compares two pseudo-MICs structurally and gives a measure of similarity between those two pseudo-MICs;

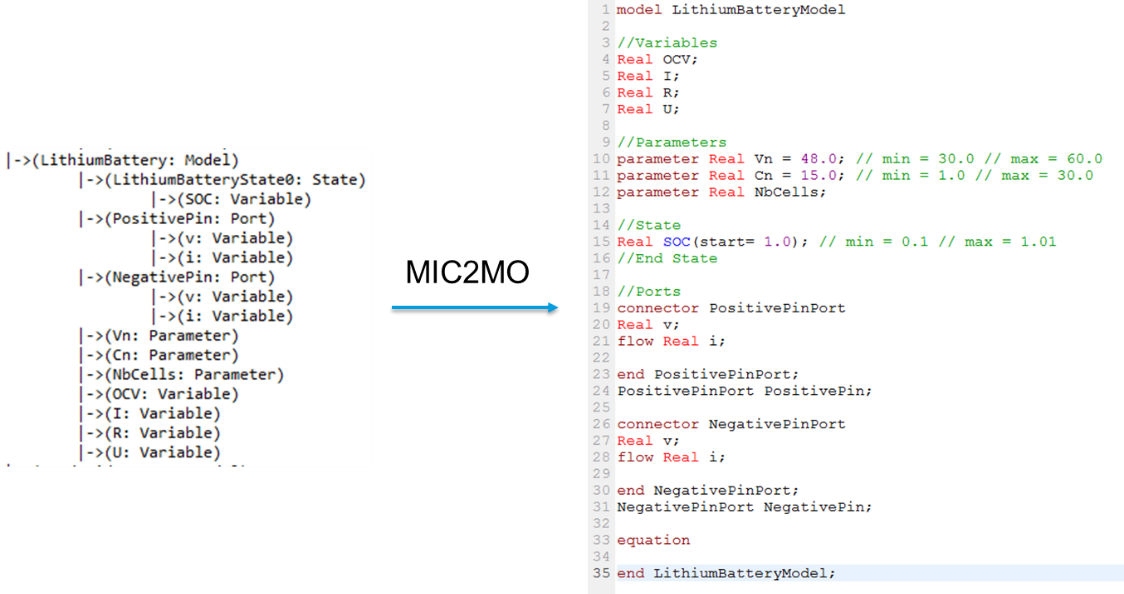
- MO2MIC: Given the code of a simulation model, it generates the pseudo-MIC of the model.

### MIC2MO

A pseudo-MIC embeds the structure of a simulation model in terms of sub-models, variables, parameters and ports. This pseudo-MIC can be interpreted recursively to generate the backbone of the simulation model. The backbone contains the declaration of the model, its sub-models, the instantiation of variables (with their possible starting values), parameters (with their value) and ports.

However, the generated backbone cannot be simulated because it has an empty equation section. No behavior is embedded in this backbone since the pseudo-MIC does not contain any information about the behavior of the simulation model. Hence, in order to be able to simulate the model, the generated backbone must be enriched by model providers with equations describing the behavior of the model.

Figure 2: Example of a MIC2MO transformation of the pseudo-MIC of a lithium battery



In Figure 2, an example of a MIC2MO transformation is given. From the pseudo-MIC of a Lithium battery model on the left, the backbone of the simulation model in Modelica code is generated on the right. As it can be seen in the code, all the variables, parameters and ports are declared. The ports are instantiated. Some variables or parameters can also be instantiated with their starting value (for variables) or value (for parameters), depending whether those properties are declared or not at the pseudo-MIC level.

### MIC2V

The mechanism MIC2V generates, given a pseudo-MIC, a program that will check requirements on simulation traces. It uses the propositional variables and requirements as defined in the pseudo-MIC specification using MITL formulas on comparisons of simulation variables to values.

Two inputs are necessary in order to run MIC2V: the pseudo-MIC of the simulation model and the file containing the results of the simulation. The pseudo-MIC embeds the definition of the requirements to be checked. The result file of the simulation is also necessary to have access to the values of the variables of interest for the requirements along the simulation run.

Modelica simulation models are considered in this work. The simulation of such models using openModelica software generates a result file in .mat format (the Matlab format). Therefore, given the definition of requirements in the pseudo-MIC and the name of the result file, a Python program is generated. It checks the MITL formulas of the requirements along the simulation run. All the operators of the MITL grammar presented above are defined. A propositional variable is defined in pseudo-MICs by comparing a simulation variable to a floating value. The propositional variable is then calculated along the simulation run by comparing the array of values of the simulation variable (from the result file) to the value. Then, the composition of logical operators combined with the use of propositional variables allows to calculate the truth of the requirements along the simulation run.

A future evolution of MIC2V could be the generation of a formatted report providing the violations of requirements. Such a kind of report could be interpreted automatically.

### CompMICs

CompMICs is a mechanism that compares pseudo-MICs structurally and gives a measure of similarity between those two pseudo-MICs.

In this work, pseudo-MICs are interpreted as graphs. Hence, comparing pseudo-MICs amounts to comparing graphs. The graph edit distance is a metric that is used to compare graphs. Comparing graph with graph using the graph edit distance gives a measure of similarity between and . The aim of the graph edit distance method is to find the simplest manner to transform into , by transforming nodes of into nodes of , by deleting nodes of or by adding nodes of according to some predefined rules that associate a cost to each unitary transformation. The cost of a global transformation from to is the sum of the costs of the unitary transformations that constitute the global transformation. The graph edit distance between and is the minimal cost of a global transformation between and . This value is influenced by the choice of the cost matrices.

Given two pseudo-MICs and , CompMICs transforms into a graph and into a graph and then return the graph edit distance between and .

### MO2MIC

The mechanism MO2MIC generates the pseudo-MIC from the simulation model. MO2MIC is in fact a compiler from Modelica code to pseudo-MICs. The Modelica grammar in EBNF (Extended Backus-Naur Form) is available in the specification of the language. It is used to create a lexer and a parser to interprete the Modelica code into an AST (Abstract Syntax Tree). This AST is then transformed into a pseudo-MIC resolving some dependencies created by the object oriented paradigm of Modelica.

The pseudo-MICs generated by the MO2MIC mechanism only contain information that is directly found in Modelica models. However, additional information that is not directly embedded in the code of simulation models can be declared at the pseudo-MIC level. This feature of the MO2MIC mechanism has to be kept in mind. For example, the definition of requirements or the properties linked to timesteps will not be found in pseudo-MICs generated by MO2MIC.

## Implementation

The mechanisms were implemented in Python. The list of files to be used is the following:

- ModelElement.py: it defines the hierarchical structure of pseudo-MICs and the different types of properties associated to each element. It is used to instantiate pseudo-MICs;

- MOLex.py: it defines the lexer for Modelica code;

- MOYacc.py: it defines the parser for the Modelica grammar;

- MICGenerationFromMO: it defines functions to generate pseudo-MICs from Modelica files and folders of Modelica files;

- MOASTToMIC: it defines functions to transform the AST produced by the parser into pseudo-MICs;

- graphRepresentation.py: it defines the structure of a graph, and functions to transform pseudo-MICs into graphs and to calculate graph edit distance;

- UnitParsing.py: it defines a parser and a lexer to transform units in Modelica into a format that is interpretable for pseudo-MICs;

- MICXMLTransformations.py: it defines functions to allow the import and export of pseudo-MICs in XML format;

- MICSetGeneration.py: it defines a function to generate a set of XML files from a set a pseudo-MICs, respecting their hierarchical structure;

- compareMICToMICs.py: it defines a function that is used to compare a pseudo-MIC to the pseudo-MICs that are in one folder;

## Conclusion

Four mechanisms were introduced in this section in order to partially automate the Modelling and Simulation process. MIC2MO generates the backbone of a simulation architecture. MIC2V creates a program to automatically check requirements on simulation traces. CompMICs compares MICs. This allows to search simulation models via their pseudo-MICs in databases of models and specifications. This mechanism can be used to help populate a network of models. MO2MIC creates the MIC of a model from its source code in Modelica. It can be used to capitalize on simulation models.

Those four mechanisms, when combined, allow to generate a network of models, to populate it and to check requirements on its simulation. They were used on the Renault case study of the SIM project (on-board electrical network).

# Case study: Heterogeneous Robotic Swarms

## Introduction

The Heterogeneous Robotic Swarms case study concerning the optimization of the on-board electric network of a mild hybrid vehicle was developed.

HARS-EvalOps: An Operational Scenarios Benchmark to support Heterogeneous Autonomous Robot Swarms innovation roadm

The four automation mechanisms presented in the previous section were used on the Renault case study.

|  |  |
| --- | --- |
| **Reference** | **Name & Description** |
| HARS\_001 | ***Mass* Attack** : A HARS advances toward an enemy concentration, dispersing along random trajectories. This HARS is composed of ground robots, air robots, and decoys. Once within range of the enemy, it deploys its weapons.  Operational metrics : attack cost, non-decoy robot attrition ratio, temporal and spatial precision of concentration, and enemy force neutralization ratio. |
| HARS\_002 | **Ambush** : A HARS observes an area around an anticipated enemy axis of progression to identify camouflaged positions, then prepositions itself and goes on standby. Passive sensors detect the passage of enemy forces, then an attack is launched.  Operational metrics : attack cost, robot attrition ratio, HARS is not detected, maximum time spent in standby mode before energy recharge, enemy force neutralization ratio. |
| HARS\_003 | **Establish a defensive position** : A HARS observes an area where the enemy must be stopped, to identify firing positions, fallback positions, and axes to mine, then prepositions itself and goes on standby. The HARS initiates combat by combining direct and indirect fire, detonating concealed explosives, and remote-controlled ammunition. It gradually withdraws to the identified positions and regularly assesses enemy losses, which it reports to a distant human command post.  Operational metrics : cost of defense, attrition ratio achieved on the enemy, duration during which the enemy is delayed. |
| HARS\_004 | **Infiltrate a HARS behind enemy lines** : A HARS infiltrates on a moonless night behind previously reconnoitered enemy lines, heads to an area of interest, and identifies high-value enemy targets. It attacks these positions, possibly with the support of distant artillery forces to which it sends firing coordinates. It reports enemy losses and shares visual evidence of their neutralization with a distant human command post. The HARS is fully expendable.  Operational metrics : non-detection of HARS, effective value of neutralized targets, number of neutralized targets. |
| HARS\_005 | **Reconnaissance in Force** : A HARS moves to an area of interest, possibly with enemies present, and observes this area to identify objects of interest. If the HARS is attacked, it retaliates against the enemies while it completes intelligence gathering and sends it to a distant human command post.  Operational metrics : HARS attrition ratio, observed surface area, number of objects of interest identified. |
| HARS\_006 | **Advancing along a logistics axis** : A ground robotic logistics convoy advances along a previously reconnoitered logistics axis. Aerial drones reconnoiter the axis of progression looking for obstacles, IEDs, or enemies. Ground robots clear these obstacles and neutralize these IEDs. If unsuccessful, the convoy determines a new route to the resupply point. Aerial drones cover the flanks and rear to anticipate threats. The HARS can neutralize enemies to protect the convoy.  Operational Metrics : Ground logistics convoy attrition. average convoy speed, average dwell time to clear each obstacle or neutralize each detected IED. Maximum mass and volume of obstacles that can be cleared. |
| HARS\_007 | **Advancing on an unrecognized axis** : A HARS advances through unrecognized terrain. Aerial drones map ahead of the rest of the swarm and suggest the best path for the ground robots, while detecting obstacles, IEDs, or enemies. Aerial drones cover the flanks and rear to anticipate threats. The HARS can neutralize enemies to protect its advance.  Operational metrics : Average convoy speed, robot attrition ratio, average stopping time to clear each obstacle or neutralize each detected IED. Maximum mass and volume of obstacles that can be cleared. Total HARS energy expenditure. |

|  |  |
| --- | --- |
| **Reference** | **Name & Description** |
| HARS\_008 | **Providing medical evacuation** : A ground robot carries a wounded person and a nurse. It moves along a previously identified axis until it reaches its point of care by the medical chain. It is accompanied by aerial drones that detect obstacles, IEDs, or enemies. If a threat is detected, the ground robot is alerted and hides.  Operational metrics : stability of the injured person's condition during transport, average speed of the ground robot, duration of alert notice, time taken to install the injured person on the platform. |
| HARS\_009 | **Deceiving the adversary** : A HARS moves to an area of interest. There, it executes the decoy scenario defined by the command: simulation of a logistical movement, defensive or offensive preparations, radio traffic between human combatants, reproduction of the electronic signature of a land or air combat vehicle.  Operational metrics : time required to implement the decoy scenario on the HARS, maximum duration of the decoy activity taking into account energy consumption, compliance of the decoy with the specification. |
| HARS\_010 | **Intercept a drone swarm** : A HARS positions itself on the axis of advance of an enemy drone swarm threatening friendly targets. It detects, locates, prioritizes, and neutralizes enemy drones, and reports the threat status to a distant human command post.  Operational metrics : duration of the detection - localization - prioritization - neutralization of enemy drones loop, attrition ratio of enemy drones. |
| HARS\_011 | **Troubleshooting a Ground Robot** : A HARS moves toward a ground robot that has reported a problem. The HARS can unstuck, tow, or restore the ground robot to its wheels. If the robot cannot be recovered or repaired, the HARS proceeds to destroy it.  Operational metrics: The ground robot is repaired, recovered or destroyed. No HARS robots are damaged in the operation. |
| HARS\_012 | **Accompanying an air combat raid** : A HARS is airlifted for a raid on an infrastructure. Aerial drones are dropped in flight to accompany the assault (decoys, remotely-controlled ammunitions) and ground robots are dropped directly onto the combat zone to infiltrate the infrastructure.  Operational metrics : time to deploy ground robots, ability to penetrate infrastructure (doors, stairs, tunnels), proportion of decoys targeted, number of targets neutralized by HARS. |
| HARS\_013 | **Accompanying the advance of a combat group** : A HARS accompanies a dismounted or embarked human combat group. This HARS splits into several groups to reconnoiter ahead of the group, protecting the flanks and rear. The HARS provides augmented vision to human combatants, particularly their commander, to anticipate threats. In the event of an attack, the HARS engages combat automatically, but can be taken over by the commander.  Operational metrics : Augmented vision range (in km), Average speed of the combat group, Total HARS energy expenditure, IED detection rate ahead of the convoy. |
| HARS\_014 | **Monitoring a sector** : A HARS deploys in a sector complementary sensors. The captured data is aggregated to a decision node robot that ensures data fusion and the regular sending of an updated tactical situation to a distant human command post.  Operational Metrics : Detection range of the deployed sensor array (at this range, any object of interest is detected by at least one sensor with 99% certainty, regardless of its direction), the link with the command post is never interrupted for more than 1 minute. Duration of surveillance without energy recharge. |
| HARS\_015 | **Covering a Combat Group's Withdrawal** : A HARS establishes a temporary defensive position in several curtains behind which a group of human combatants in contact can withdraw. The HARS initiates combat by combining direct fire, indirect fire, and remotely operated munitions.  Operational metrics : length of time enemy is delayed by swarm, attrition gained on enemy, no friendly fire. |
| HARS\_016 | **Relaying Communication Signals** : A HARS positions itself on a high ground or any area favorable to radio communication relays. It relays communications between groups of human combatants and a distant command post beyond its range.  Operational metrics : robustness of the link to jamming and weather conditions, extension of the communication range offered to human combatants, maximum mission duration taking into account the energy consumption of the HARS, bandwidth offered. |
| HARS\_017 | **Jamming signals** : The HARS positions itself in an area where enemy communication signals are passing. It picks up and analyzes these signals, then establishes a jamming strategy, and finally transmits its own jamming signal towards the enemy transmitter and receiver.  Operational metrics : HARS is not detected due to its jamming signals, jamming effectiveness, jamming range. |
| HARS\_018 | **Proactively decide** : The HARS receives mission parameters as input, which can be modified by human command during the mission. It has the ability to plan the rest of its mission with on-board or external computing capabilities (access to a computing server in an infrastructure or in a combat vehicle): it determines the optimal configurations of the swarm for the mission, defines sequences of actions to be executed, and the decision rules in case of events, taking into account the swarm's energy budget.  Operational metrics : minimized energy consumption for plan development, consistency of plans with mission parameters. |
| HARS\_019 | **Seizing a fortified enemy position in collaboration with human combatants** : A HARS locates the enemy position and identifies axes of advance for the ground robots and the human combat group, as well as exploitable cover and firing positions. The HARS advances towards the enemy position, some of the robots being decoys. The HARS strikes the enemy with direct fire, remotely-controlled ammunitions, or by sending firing coordinates to indirect support from the human combat group. The human combat group also advances towards the enemy position, concentrating on the most dangerous targets.  Operational metrics : Attack cost, HARS attrition ratio during advance towards the enemy, HARS/human combat group speed for detection-identification-targeting-indirect fires loop, human combat group attrition ratio. |
| HARS\_020 | **Providing fire support to a human combat group** : A HARS provides direct, indirect, and remotely operated munitions fire support to a human combat group in contact. Targeting is provided by either human combatants or aerial robots. Fire support allocation can be provided by the human commander or a HARS decision node.  Operational metrics : HARS / human combat group speed for detection-identification-targeting-indirect fires loop. |
| HARS\_021 | **Pursuing a fleeing enemy** : A HARS pursues an enemy, whether mounted or dismounted, providing targeting data for robotic or human fire support, and striking it with remotely-controlled ammunitions.  Operational metrics : Robot speed sufficient to pursue enemy, attrition achieved on enemy, HARS / human combat group speed for detection-identification-targeting-indirect fire loop. |
| HARS\_022 | **Managing a surrendering enemy force** : An enemy signals its willingness to cease combat within detection range of a HARS. The HARS transmits a video and audio feed to a human command post and broadcasts the human commander's surrender instructions. The HARS observes enemies and reports abnormal behavior. The HARS positions visible armed robots to materialize a threat, under strict human command control.  Operational metrics : compliance with the laws of war, all detected enemies remain within the detection field of the HARS sensors throughout the surrender phase. |

## Conclusion

The original optimization of the Renault case study was performed except that the geometrical constraint was solved manually prior to the optimization.

# Conclusion

The Model Architect Problem was treated within this report. Multi criteria scheduling was presented and the system of notation of scheduling problems (three fields notation) was introduced. This notation was used to express the Model Architect Problem in terms of multi criteria scheduling. However, different definitions of the MAP can be used, depending on the viewpoint concerning the relationships between resources and tasks.

# Références / Bibliographie

Balci, O. (2004). Quality assessment, verification, and validation of modeling and simulation applications. *Simulation Conference, 2004. Proceedings of the 2004 Winter*.

Beiglböck, M. a.-K. (2012). Utility maximization, risk aversion, and stochastic dominance. *Mathematics and Financial Economics*, 1--13.

Ben-Haim, Y. a. (2012). Robustness, fidelity and prediction-looseness of models. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Science*, 227--244.

Chaves, A. a. (2014). Comparison of risk analysis approaches and a case study of the risk of incorporating solar photovoltaic systems into a commercial electric power grid. *Systems Engineering*.

Chen, S. H. (2007). A Comparative Study of Uncertainty Propagation Methods for Black-Box Type Functions. *ASME 2007 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*.

Chen, W. a. (2004). Analytical uncertainty propagation via metamodels in simulation-based design under uncertainty.

Cousot, P. a. (1992). Abstract interpretation frameworks. *Journal of logic and computation*, 511--547.

Crnkovic, I. a. (2004). Classification of quality attributes for predictability in component-based systems. *Proc. of Workshop on Architecting Dependable Systems*.

Fontaine, G. a. (2016). Multi-objective Optimization of Automotive Electrical/Energy Storage System. *IEEE International Conference on Industrial Technology (ICIT2016)*.

Fontaine, G. (s.d.). *Theoretical modeling and associated processes for Model Architects in a multidisciplinary environment. Palaiseau, Ecole Polytechnique. Ongoing.*

Georgescu, I. (2012). Expected utility operators and possibilistic risk aversion. *Soft Computing*, 1671--1680.

Gross, D. C. (1999). *Report from the fidelity implementation study group.*

Küttler, S. (2013). *Dimensionnement optimal de machines synchrones pour des applications de véhicules hybrides. PhD Thesis. Compiègne: Université de Technologie de Compiègne.*

Malak, R. a. (2009). Modeling design concepts under Risk and Uncertainty using Parameterized Efficient Sets. *SAE International Journal of Materials and Manufacturing*, 339-352.

Milanese, M. a. (2003). Model quality in nonlinear SM identification. *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*, 6021--6026.

NASA. (2008). *NASA Standard for Models and Simulations.*

Oberkampf, W. L. (2002). Error and uncertainty in modeling and simulation. *Reliability Engineering & System Safety*, 333--357.

Page, E. H. (1999). Observations on the complexity of composable simulation. *Proceedings of the 31st conference on Winter simulation: Simulation---a bridge to the future-Volume 1*, 553--560.

Paredis, B. D. (2010). A Review of Methods for Design under Uncertainty from the Perspective of Utility Theory. *ASME 2010 International Design Engineering Technical Conf. \& Computers and Information in Engineering Conf.*

Petty, M. D. (2003). A composability lexicon. *Proceedings of the Spring 2003 Simulation Interoperability Workshop*, 181--187.

Petty, M. D. (2003). A formal basis for a theory of semantic composability. *Proceedings of the Spring 2003 Simulation Interoperability Workshop*, 416--423.

Petty, M. D. (2003). Computational complexity of selecting components for composition. *Proceedings of the Fall 2003 Simulation Interoperability Workshop*, 14--19.

Pinedo, M. (1995). *Scheduling: theory, algorithms, and systems. 1995.* Prentice-Hall, Englewood Cliffs, NJ.

Radhakrishnan, R. a. (2005). A methodology for model selection in engineering design. *Journal of mechanical design*, 378--387.

Ruderman, A. M.-K. (2013). Simulation-based conjoint ranking for optimal decision support process under aleatory uncertainty. *Journal of Intelligent Manufacturing*, 641--652.

Stevenson, D. (2005). A Critical Look at Design, Verification, and Validation of Large Scale Simulations.

T'kindt, V. a.-C. (2006). *Multicriteria scheduling: theory, models and algorithms.* Springer.

Van Bossuyt, D. L. (2013). A case for trading risk in complex conceptual design trade studies. *Research in Engineering Design*, 259--275.

Van Buren, K. L. (2014). Model selection through robustness and fidelity criteria: Modeling the dynamics of the CX-100 wind turbine blade. *Mechanical Systems and Signal Processing*, 246--259.

Von Neumann, J. a. (1944). *Theory of Games and Economic Behavior.*

Walker, W. E. (2003). Defining uncertainty: a conceptual basis for uncertainty management in model-based decision support. *Integrated assessment*, 5--17.

Zadeh, L. A. (2006). Generalized theory of uncertainty (GTU)—principal concepts and ideas. *Computational Statistics & Data Analysis*, 15--46.

Zhang, S. a. (2013). Concurrent treatment of parametric uncertainty and metamodeling uncertainty in robust design. *Structural and Multidisciplinary Optimization*, 63--76.

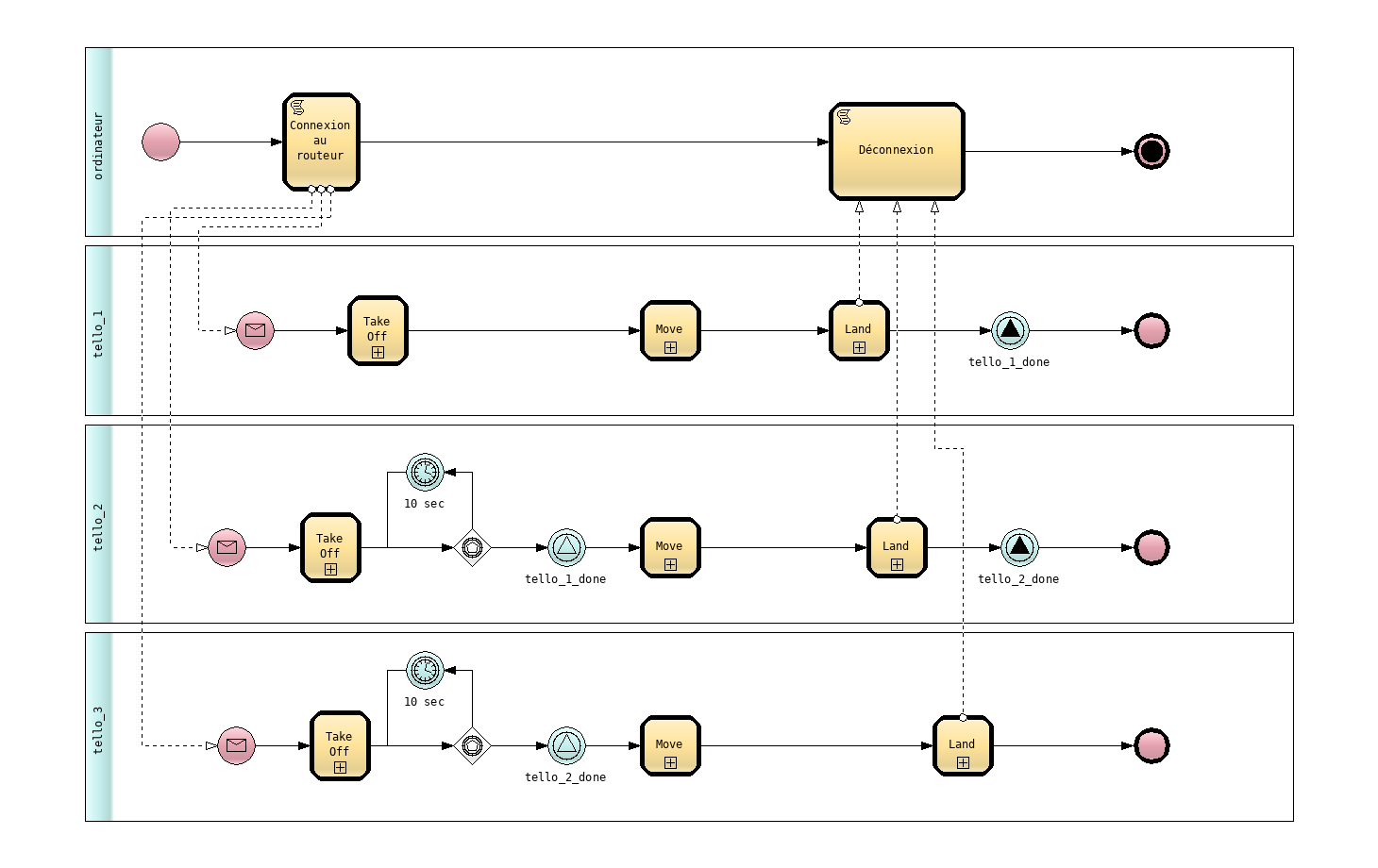
# Annexes

# Annex 1 – Benchmark modeling with BPMN

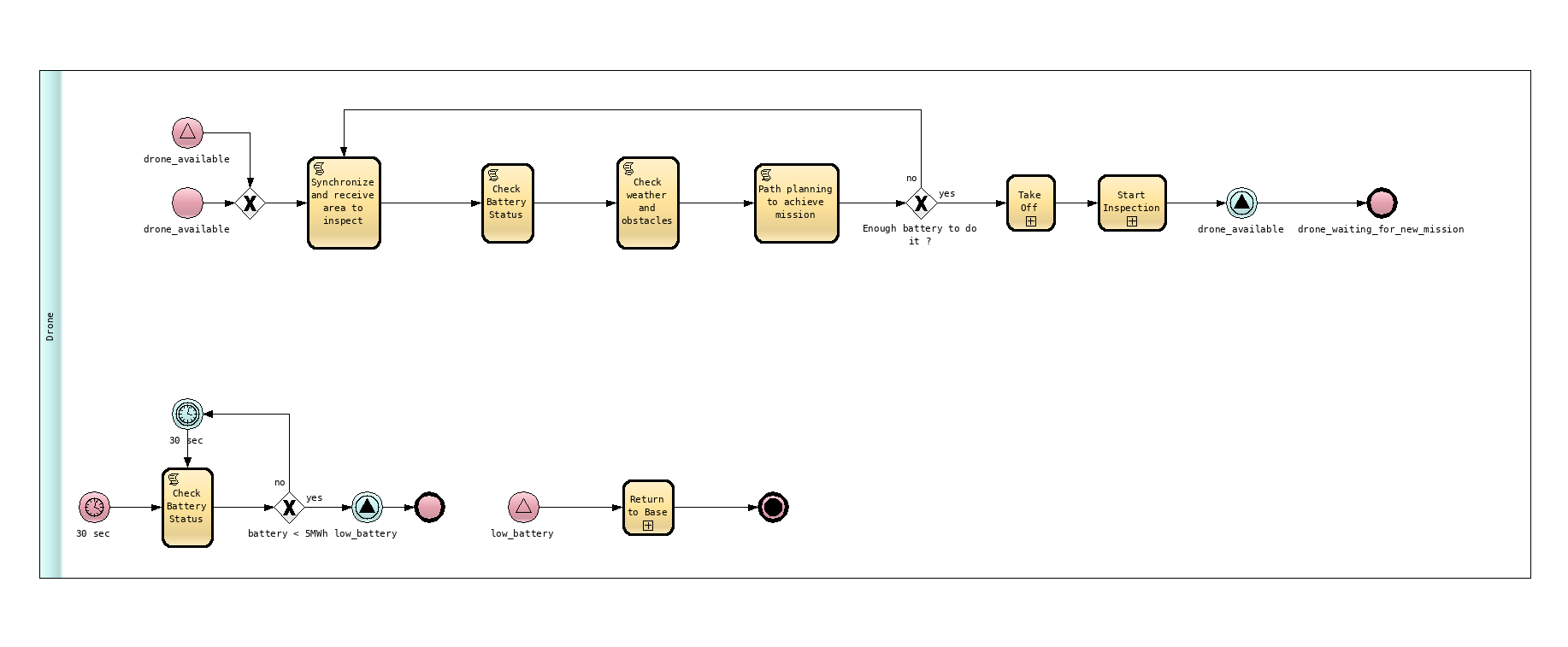
|  |  |  |  |
| --- | --- | --- | --- |
| Benchmark | Description | UAV | Husky |
| B1 | Test operation of 3 Tello EDU UAVs in parallel | 3 |  |
| B2 | BPMN of a UAV drone for a collaborative mission to inspect the structure of ENSTA's S building | 3/4 |  |
| B3 | Battery consumption verification scenario for window frame contour analysis | 1 |  |
| B4 | Scenario for checking the battery consumption of a DJI Tello EDU drone making circular turns around a red dot on an indoor wall. | 1 |  |
| B5 | Linear inspection scenario of a wall in a corridor to detect cracks | 1 |  |
| B6 |  |  |  |
|  |  |  |  |

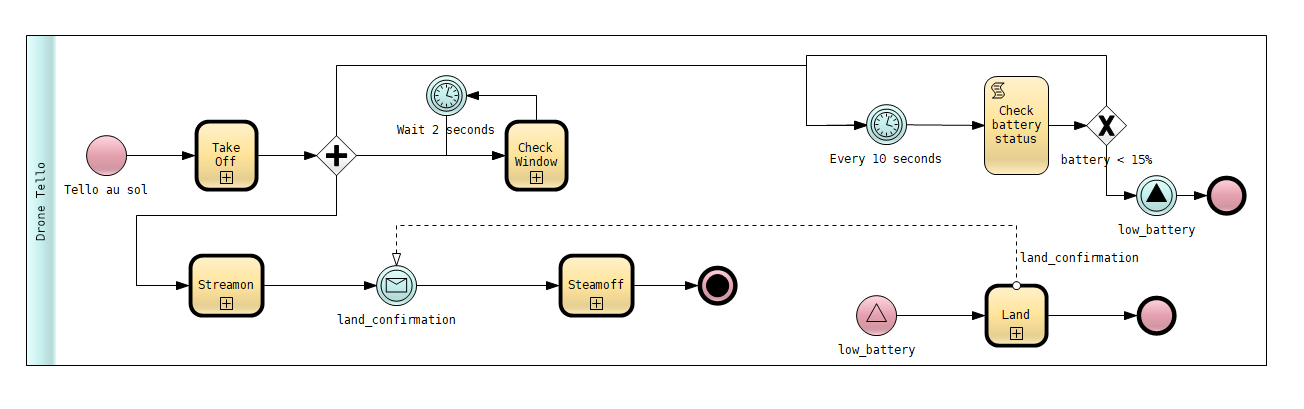


B1 – Test operation of 3 Tello EDU UAVs in parallel.

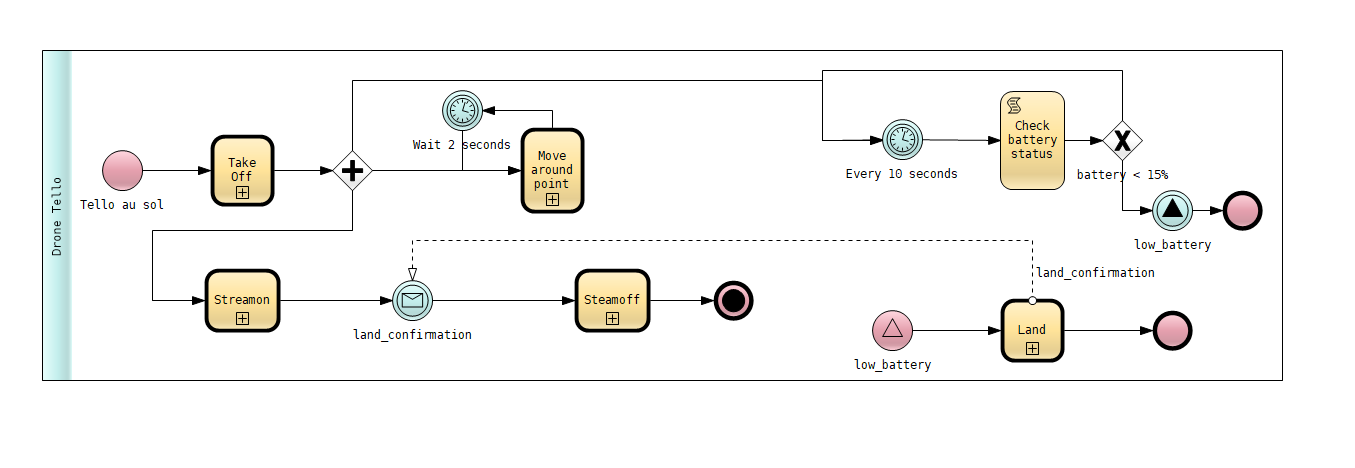


B2 – BPMN of a UAV drone for a collaborative mission to inspect the structure of ENSTA's S building.

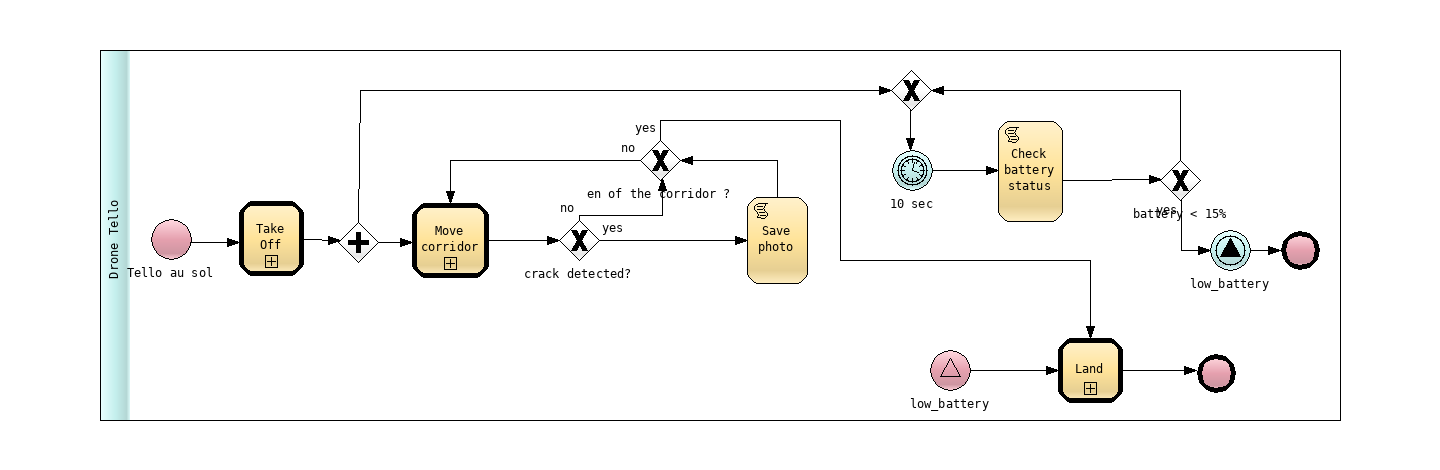


B3 – Scenario for checking the battery consumption of a DJI Tello EDU drone for contour analysis of a window frame (80 ± 2 cm wide, 180 ± 2 cm high) indoors.

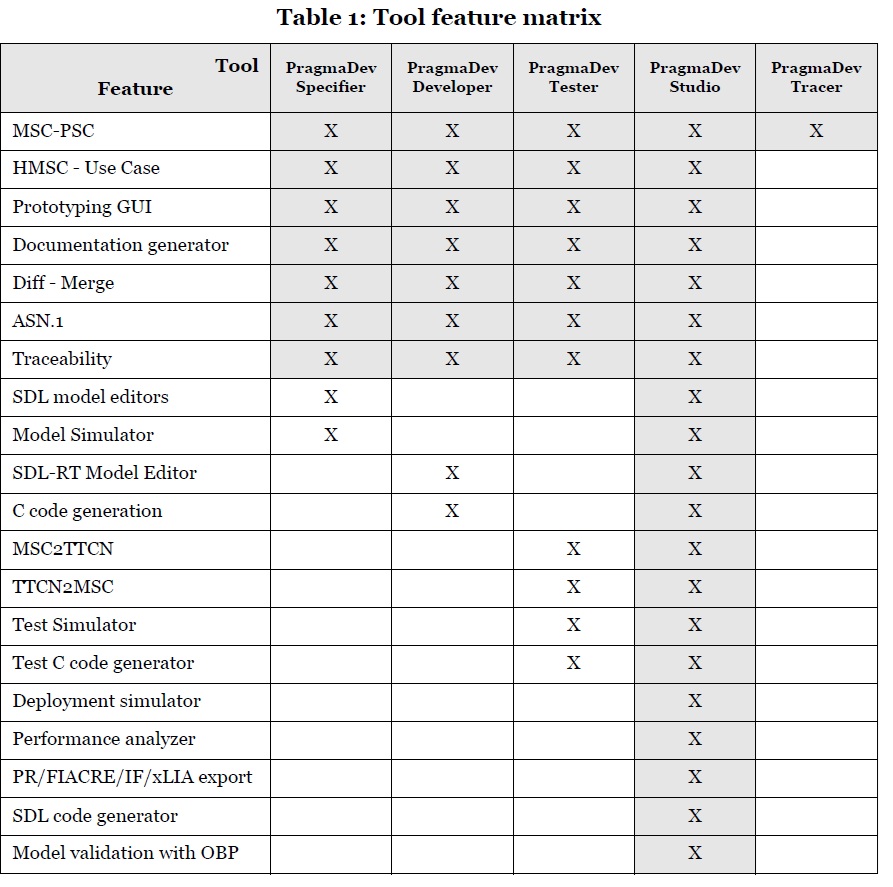
B4 – Scenario for checking the battery consumption of a DJI Tello EDU drone making circular turns around a red dot on an indoor wall.

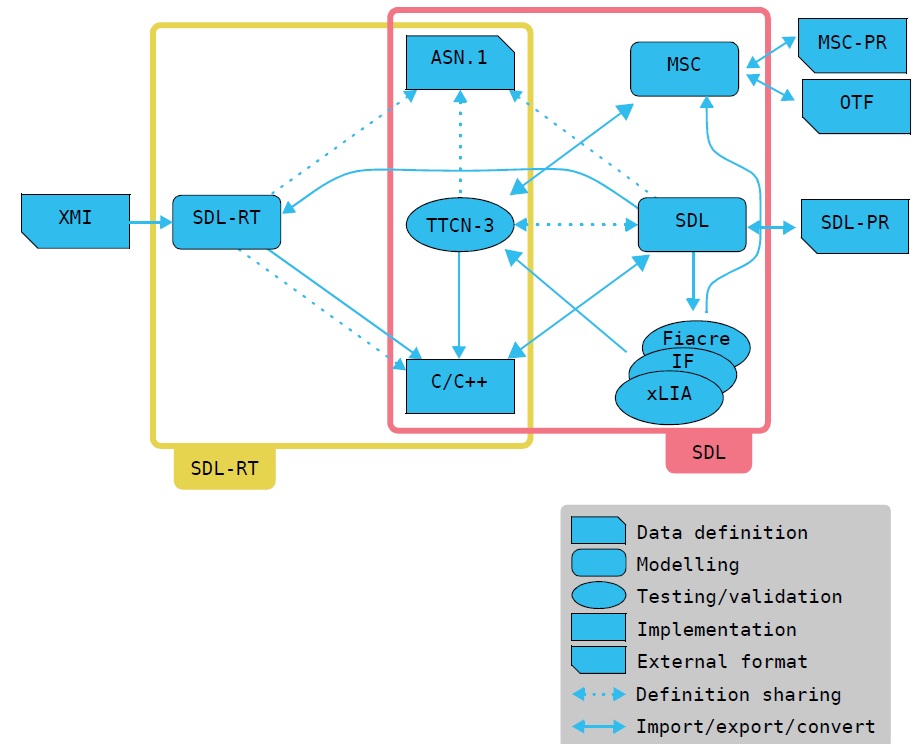


B5 – Linear inspection scenario of a wall in a corridor to detect cracks. As each crack is detected, the DJI Tello EDU drone records a photo, then continues its trajectory to the end of the wall, or until the battery reaches 15%.



# Annex 2 – Pragmadev





There are a multitude of software packages like **PragmaDev Process** for modeling **BPMN** processes as presented above, but what sets it apart from other BPMN software is its ability to perform formal **model checking**, thanks in particular to automatic generation of executable models and the integration of model-checking techniques. This makes it possible to identify errors such as **deadlocks**, undesired behavior, or violations of temporal properties (**LTL - Linear Temporal Logic**), which few standard BPMN tools offer. PragmaDev Process is particularly well suited to mission-critical systems (aerospace, defense, industrial automation, etc.) where process reliability is paramount.

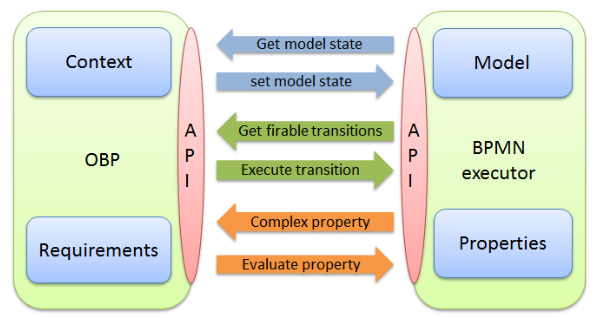
There are two key features for validating a BPMN model: **exploration** and **simulation**, each with a specific purpose and function :

* ***Exploration (Model-checking)***

**Exploration** uses a formal engine (**OBP** - One.Bug.Process) to automatically run through all possible process execution scenarios. It relies on algorithms such as Nested-DFS to check logical properties (often expressed in **LTL, Linear Temporal Logic**), such as:

* "Does this process always finish?"
* "Is a certain state reached in all executions?"
* "Are there any possible deadlocks?”

Exploration is **exhaustive**: it explores all path combinations in the model, which makes it very powerful for detecting errors, but potentially slower.



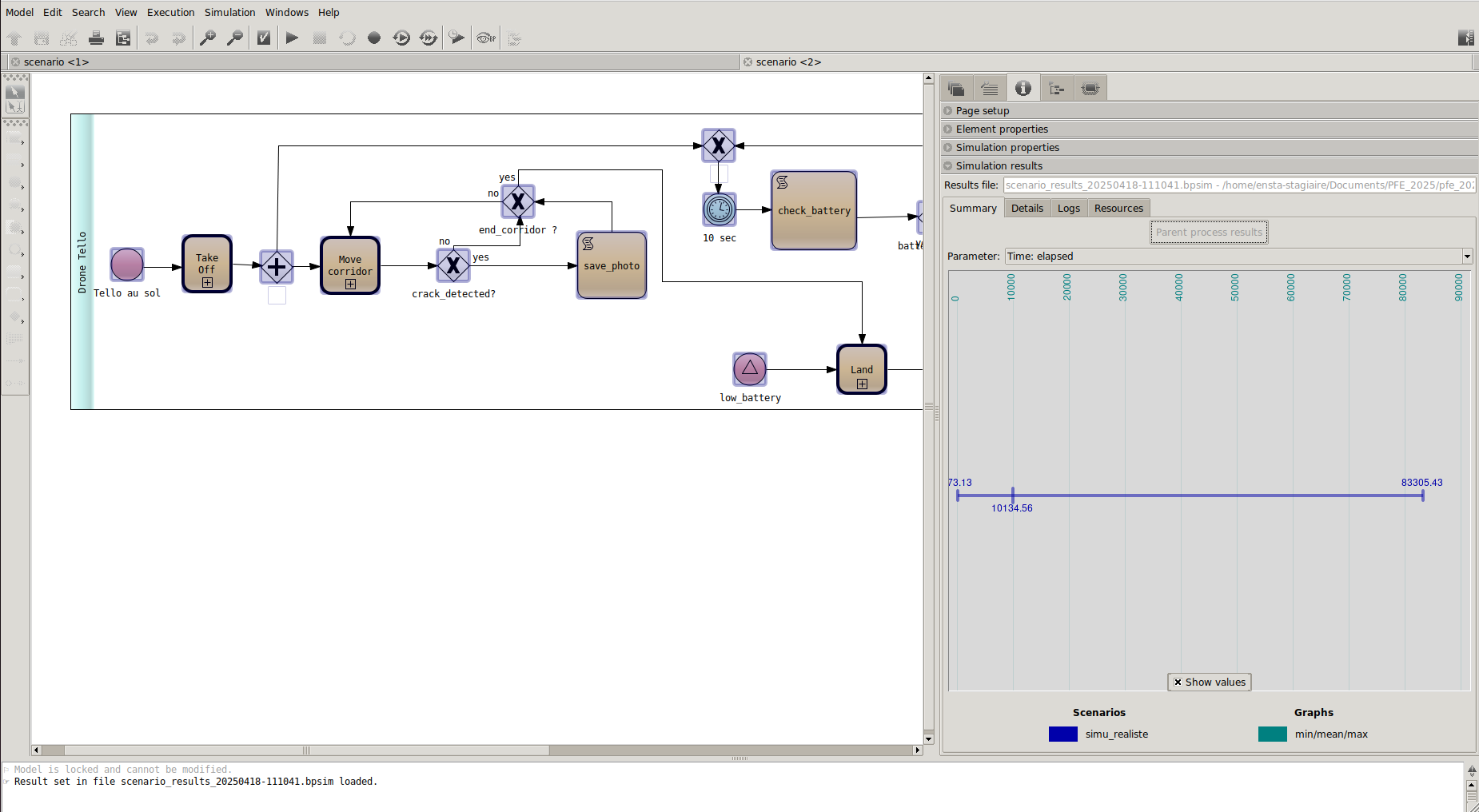
* ***Simulation***

**Simulation** can be used to manually or automatically test a specific path in the process. It is used to **dynamically** observe **model behavior**, for the purposes of understanding, testing or debugging. Unlike exploration, it does not guarantee complete coverage: only tested scenarios are observed.

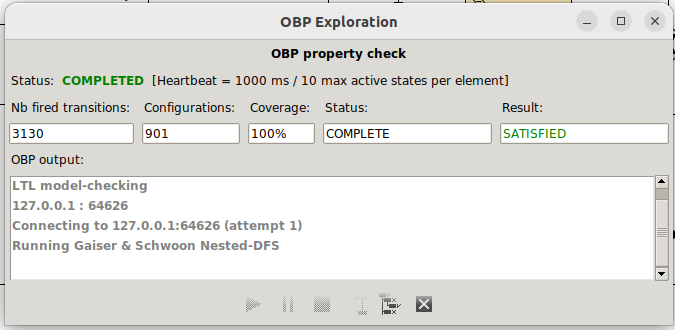
During simulation, the user can :

* Trigger events manually,
* Choose paths among gateways,
* Observe changes in the state of tasks and variables.

Exemple :



Here you can see the **estimated time** needed (minimum, mode & maximum) to complete the analysis of the corridor in scenario B5, and you can also explore the model to check for complexity and possible blockages (deadlocks) :



The window displayed corresponds to an OBP (One.Bug.Process) scan result, which is the built-in verification engine. Here are the key parameters of this window:

* **Status: COMPLETED**: indicates that the analysis has been successfully completed.
* **Heartbeat** : control parameters, the heart beats every second to monitor the analysis, and each element can have up to 10 simultaneous active states.
* **Nb fired transitions**: total number of transitions (changes of state) simulated during analysis.
* **Configurations**: total number of unique states explored.
* **Coverage**: all model elements were covered during exploration.
* **Result: SATISFIED**: indicates that the specified LTL property has been satisfied (no violations detected).
* **OBP output** (text box): contains session logs, here showing that the engine used the Gaiser & Schwoon Nested-DFS algorithm for LTL model-checking, and connected to the local server via port 64626.

# Annex 3 – LTL – Linear Temporal Logic – Model Checking

In [logic](https://en.wikipedia.org/wiki/Logic), linear temporal logic or linear-time temporal logic[[1]](https://en.wikipedia.org/wiki/Linear_temporal_logic" \l "cite_note-1)[[2]](https://en.wikipedia.org/wiki/Linear_temporal_logic" \l "cite_note-2) (LTL) is a [modal](https://en.wikipedia.org/wiki/Modal_logic) [temporal logic](https://en.wikipedia.org/wiki/Temporal_logic) with modalities referring to time. In LTL, one can encode [formulae](https://en.wikipedia.org/wiki/Formula_(logic)) about the future of [paths](https://en.wikipedia.org/wiki/Path_(graph_theory)), e.g., a condition will eventually be true, a condition will be true until another fact becomes true, etc. It is a fragment of the more complex [CTL\*](https://en.wikipedia.org/wiki/CTL*), which additionally allows branching time and [quantifiers](https://en.wikipedia.org/wiki/Quantifier_(logic)). LTL is sometimes called propositional temporal logic (PTL).[[3]](https://en.wikipedia.org/wiki/Linear_temporal_logic" \l "cite_note-Gabbay2003-3) In terms of [expressive power](https://en.wikipedia.org/wiki/Expressive_power_(computer_science)), LTL is a fragment of [first-order logic](https://en.wikipedia.org/wiki/First-order_logic).[[4]](https://en.wikipedia.org/wiki/Linear_temporal_logic" \l "cite_note-4)[[5]](https://en.wikipedia.org/wiki/Linear_temporal_logic" \l "cite_note-5)

LTL was first proposed for the [formal verification](https://en.wikipedia.org/wiki/Formal_verification) of computer programs by [Amir Pnueli](https://en.wikipedia.org/wiki/Amir_Pnueli) in 1977

Syntax

LTL is built up from a finite set of [propositional variables](https://en.wikipedia.org/wiki/Propositional_variable) *AP*, the [logical operators](https://en.wikipedia.org/wiki/Logical_connective) ¬ and ∨, and the [temporal](https://en.wikipedia.org/wiki/Temporal_logic) [modal operators](https://en.wikipedia.org/wiki/Modal_operator) X (some literature uses O or N) and U. Formally, the set of LTL formulas over *AP* is inductively defined as follows:

* if *p* ∈ *AP* then *p* is an LTL formula;
* if *ψ* and *φ* are LTL formulas then ¬*ψ*, *φ* ∨ *ψ*, X *ψ*, and *φ* U *ψ* are LTL formulas.[[7]](https://en.wikipedia.org/wiki/Linear_temporal_logic" \l "cite_note-7)

X is read as next and U is read as until. Other than these fundamental operators, there are additional logical and temporal operators defined in terms of the fundamental operators, in order to write LTL formulas succinctly. The additional logical operators are ∧, →, ↔, true, and false. Following are the additional temporal operators.

* G for always (globally)
* F for finally
* R for release
* W for weak until
* M for mighty release

Semantics

An LTL formula can be [*satisfied*](https://en.wikipedia.org/wiki/Satisfiability) by an infinite sequence of truth valuations of variables in *AP*. These sequences can be viewed as a word on a path of a [Kripke structure](https://en.wikipedia.org/wiki/Kripke_structure) (an [ω-word](https://en.wikipedia.org/wiki/Ω-language) over [alphabet](https://en.wikipedia.org/wiki/Alphabet_(formal_languages)) 2*AP*). Let *w* = a0,a1,a2,... be such an ω-word. Let *w*(*i*) = *ai*. Let *w*i = *ai*,*ai*+1,..., which is a suffix of *w*. Formally, the satisfaction relation ⊨ between a word and an LTL formula is defined as follows:

* *w* ⊨ *p* if *p* ∈ *w*(0)
* *w* ⊨ ¬*ψ* if *w* ⊭ *ψ*
* *w* ⊨ *φ* ∨ *ψ* if *w* ⊨ *φ* or *w* ⊨ *ψ*
* *w* ⊨ X *ψ* if *w*1 ⊨ *ψ* (in the next time step *ψ* must be true)
* *w* ⊨ *φ* U *ψ* if there exists *i* ≥ 0 such that *wi* ⊨ *ψ* and for all 0 ≤ *k* < i, *wk* ⊨ *φ* (*φ* must remain true until *ψ* becomes true)

We say an ω-word *w* satisfies an LTL formula *ψ* when *w* ⊨ *ψ*. The [ω-language](https://en.wikipedia.org/wiki/Ω-language) *L*(*ψ*) defined by *ψ* is {*w* | *w* ⊨ *ψ*}, which is the set of ω-words that satisfy *ψ*. A formula *ψ* is *satisfiable* if there exist an ω-word *w* such that *w* ⊨ *ψ*. A formula *ψ* is *valid* if for each ω-word *w* over alphabet 2*AP*, we have *w* ⊨ *ψ*.

The additional logical operators are defined as follows:

* *φ* ∧ *ψ* ≡ ¬(¬*φ* ∨ ¬*ψ*)
* *φ* → *ψ* ≡ ¬*φ* ∨ *ψ*
* *φ* ↔ *ψ* ≡ (*φ* → *ψ*) ∧ ( *ψ* → *φ*)
* true ≡ *p* ∨ ¬*p*, where *p* ∈ *AP*
* false ≡ ¬true

The additional temporal operators R, F, and G are defined as follows:

* *ψ* R *φ* ≡ ¬(¬*ψ* U ¬*φ*) ( *φ* remains true until and including once *ψ* becomes true. If *ψ* never becomes true, *φ* must remain true forever. *ψ* releases *φ*.)
* F *ψ* ≡ true U *ψ* (eventually *ψ* becomes true)
* G *ψ* ≡ false R *ψ* ≡ ¬F ¬*ψ* (*ψ* always remains true)

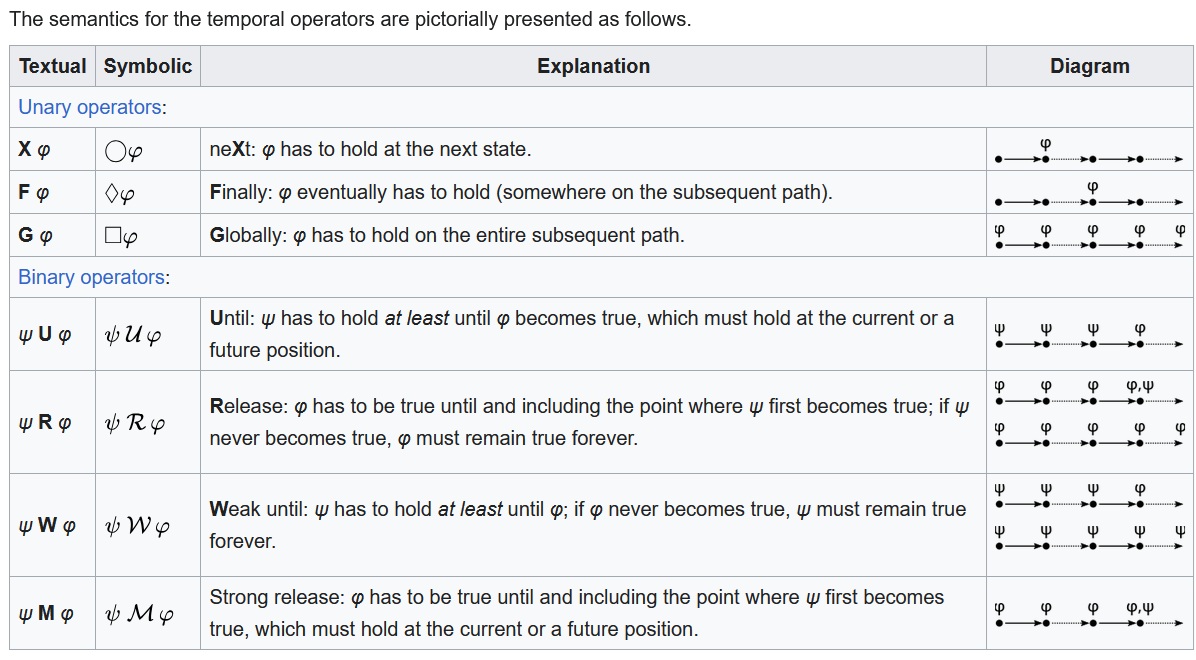
Weak until and strong release

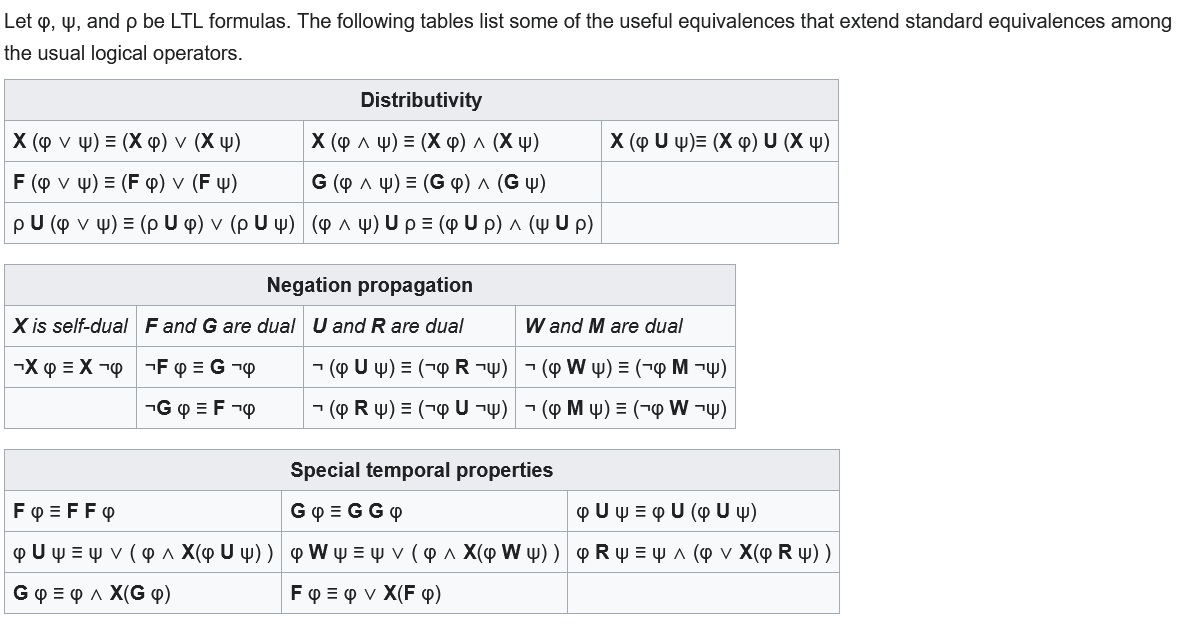
Some authors also define a *weak until* binary operator, denoted W, with semantics similar to that of the until operator but the stop condition is not required to occur (similar to release). It is sometimes useful since both U and R can be defined in terms of the weak until:

* *ψ* W *φ* ≡ (*ψ* U *φ*) ∨ G *ψ* ≡ *ψ* U (*φ* ∨ G *ψ*) ≡ *φ* R (*φ* ∨ *ψ*)
* *ψ* U *φ* ≡ F*φ* ∧ (*ψ* W *φ*)
* *ψ* R *φ* ≡ *φ* W (*φ* ∧ *ψ*)

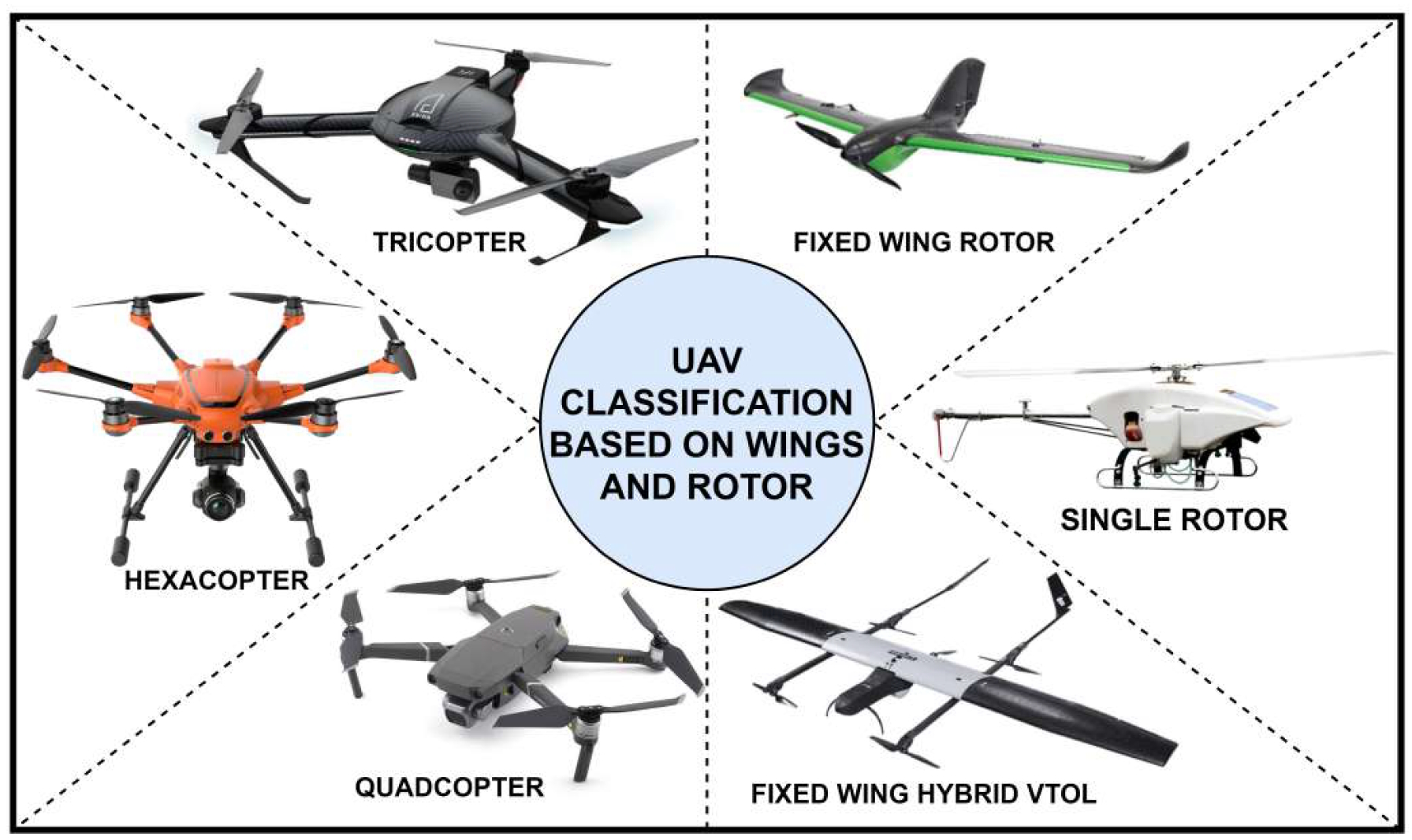
The *strong release* binary operator, denoted M, is the dual of weak until. It is defined similar to the until operator, so that the release condition has to hold at some point. Therefore, it is stronger than the release operator.

* *ψ* M *φ* ≡ ¬(¬*ψ* W ¬*φ*) ≡ (*ψ* R *φ*) ∧ F *ψ* ≡ *ψ* R (*φ* ∧ F *ψ*) ≡ *φ* U (*ψ* ∧ *φ*)





# Annex 4 – UAV – Tello – Husky

(image = [Researchgate](https://www.researchgate.net/publication/344592031_A_Comprehensive_Review_of_Unmanned_Aerial_Vehicle_Attacks_and_Neutralization_Techniques))

An **unmanned aerial vehicle** (**UAV**) or **unmanned aircraft system** (**UAS**), commonly known as a **drone**, is an [aircraft](https://en.wikipedia.org/wiki/Aircraft) with no human [pilot](https://en.wikipedia.org/wiki/Aircraft_pilot), crew, or passengers onboard, but rather is controlled remotely or is autonomous.

(definition : [UAV - Wikipedia](https://en.wikipedia.org/wiki/Unmanned_aerial_vehicle))

* **DJI Tello EDU :**

| Mode | Description |
| --- | --- |
| Wi-Fi Direct | Le drone crée son propre réseau Wi-Fi ; le PC/Smartphone s’y connecte |
| Station Mode (AP Client) | Le drone se connecte à un routeur Wi-Fi partagé (mode multi-drone) |
| Contrôle manuel | Via l’application mobile Tello/Tello EDU |
| Contrôle programmé | Via des scripts Python ou des blocs Scratch |

| Langage | Utilisation principale |
| --- | --- |
| Python | Contrôle avancé via SDK, souvent avec tello-py ou djitellopy |
| Scratch | Programmation visuelle (idéale pour débutants et enfants) |
| Swift | Utilisable sur iPad/iOS via l'app officielle Tello EDU |
| ROS/ROS2 | Pour l’intégration robotique avancée (via packages comme tello\_ros ou tello-ros2-gazebo) |

| **Catégorie** | **Détail** |
| --- | --- |
| Dimensions et Poids |  |
| Poids | Environ 87 g (avec hélices et batterie) |
| Dimensions | 98 × 92,5 × 41 mm |
| Taille des hélices | 3 pouces |
| Fonctions Intégrées |  |
| Télémètre | Oui |
| Baromètre | Oui |
| LED | Oui |
| Système de vision | Oui |
| Wi-Fi | 802.11n 2,4 GHz |
| Transmission vidéo | 720p en direct |
| Port de charge | Micro USB |
| Performances de Vol |  |
| Distance maximale de vol | 100 m |
| Vitesse maximale | 8 m/s (17,8 mph) |
| Temps de vol maximal | 13 minutes |
| Altitude maximale de vol | 30 m |
| Plage de fonctionnement du télémètre | 0,3 m à 10 m (optimal entre 0,3 m et 6 m) |
| Batterie |  |
| Type | Batterie amovible |
| Capacité | 1,1 Ah / 3,8 V |
| Spécifications | 1S 1100mAh |
| Caméra |  |
| Résolution photo | 5 MP (2592 × 1936) |
| Champ de vision (FOV) | 82,6° |
| Vidéo | HD 720p à 30 fps |
| Formats | JPG (photos), MP4 (vidéos) |
| Stabilisation | Stabilisation électronique de l’image (EIS) |
| Unité de Traitement de la Vision |  |
| VPU | Intel Movidius Myriad 2 |
| CPU | SPARC microprocessor core |
| Puissance de calcul | 80 à 150 GFLOPS pour environ 1 W |
| Nombre de cœurs | 14 |
| Fréquence | 933 MHz |
| Mémoire | 4GB LP-DDR3 (interface 32 bits à 733 MHz) |
| Températures de fonctionnement | -40 °C à 105 °C |
| Source VPU | [Lien GitHub](https://github.com/MrJabu/RyzeTelloFirmware/blob/master/README.md) |
| Moteurs | 4 moteurs brushed |

* **Robot Husky UGV :**

| Élément | Détail |
| --- | --- |
| OS | Ubuntu LTS + ROS/ROS2 préinstallé |
| Connectivité | USB, Ethernet, Wi-Fi, ports CAN, GPIO, série (RS232/485) |
| Capteurs compatibles | LIDAR, GPS, IMU, caméras ZED/RealSense, radars, RTK, etc. |
| Capteurs intégrés (de base) | IMU, encodeurs roues, alimentation monitorée |

*Dimensions et Caractéristiques Physiques*

| Élément | Détail |
| --- | --- |
| Poids | ~50 kg |
| Dimensions | 990 × 670 × 390 mm (L × l × H) |
| Charge utile | Jusqu’à 75 kg |
| Châssis | Aluminium, IP65, tout-terrain |
| Propulsion | 4 roues motrices indépendantes (differential drive) |
| Batterie | 2 × batteries SLA 12 V / 20 Ah ou Li-Ion (option) |
| Autonomie | 1 à 3 heures selon usage |
| Vitesse maximale | ~1 m/s (modulable) |

*Modes de Contrôle*

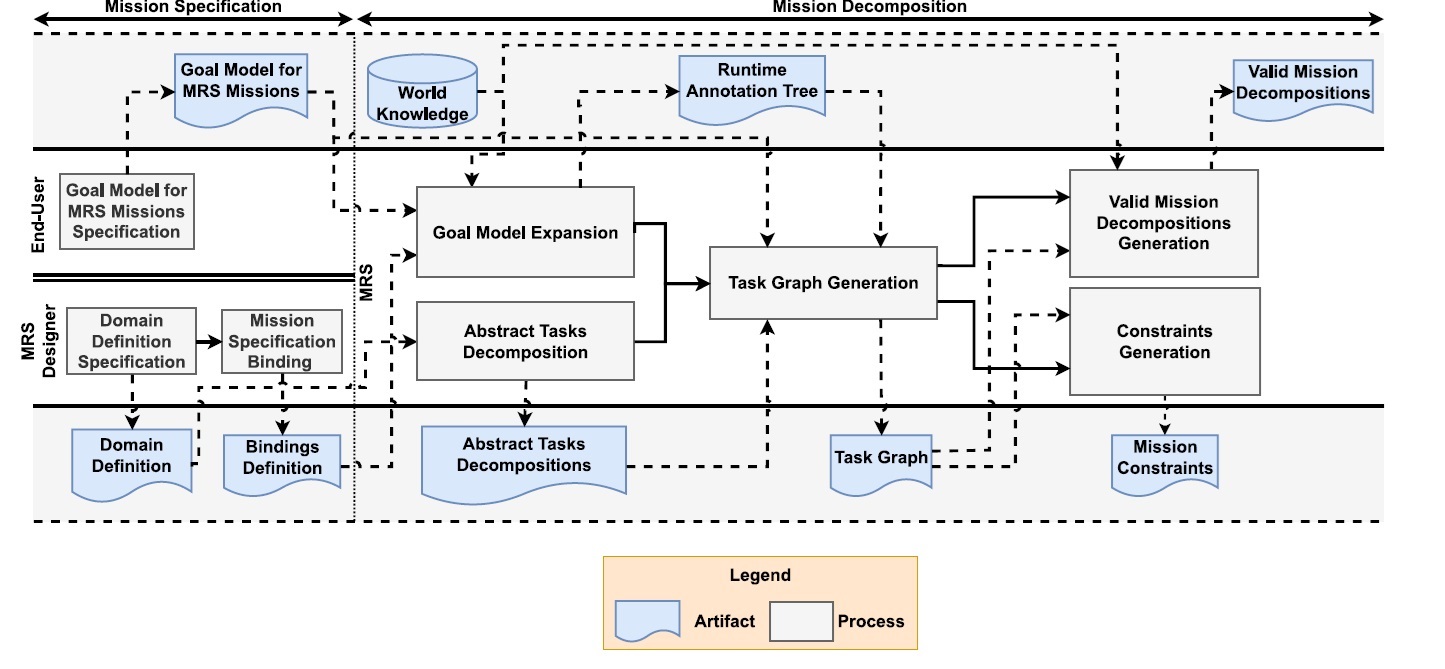
| Mode | Description |
| --- | --- |
| Manuel | Télécommande, joystick USB, ROS teleop |
| Autonome | Via ROS/ROS2 : navigation, SLAM, path planning |
| API réseau | Accès via SSH ou socket sur ROS master/slave |
| Gazebo | Simulation 3D complète dans ROS (avec package officiel husky\_gazebo) |

*Programmation et Intégration*

| Outil / Langage | Usage principal |
| --- | --- |
| ROS1 / ROS2 | Architecture principale (topics, services, actions) |
| Python / C++ | Pour les nodes ROS |
| Gazebo | Simulation et test des algos dans un monde virtuel |
| RViz | Visualisation 3D des capteurs et trajectoires |
| Move Base / Nav2 | Navigation autonome |
| SLAM | GMapping, Cartographer, RTAB-Map, etc. |

# Annex 5 – NAF v.3 – MEGA HOPEX v.5 – BPMN – HRS Modeling and Simulation

# Annex 7 – FACE



1. Etat = “Ebauche”; “En revue”; “Validé”. [↑](#footnote-ref-2)