火龙果・整理
uml.org.cn

# Modelling and Simulation for the Integrated Design of Mechatronic Systems

**Giacomo Barbieri** * **Gabriele Goldoni** ** **Roberto Borsari** ** **Cesare Fantuzzi** *

* *University of Modena and Reggio Emilia, Via Amendola 2, 42122 Reggio Emilia (Italy) {giacomo.barbieri, cesare.fantuzzi}@unimore.it*
** *Tetra Pak Packaging Solutions SpA, Via Delfini 1, 41123 Modena (Italy) {gabriele.goldoni, roberto.borsari}@tetrapak.com*

**Abstract:** The development of mechatronic systems involves the use of multiple disciplines, from mechanical engineering to electronics engineering and computer science. Traditionally, every discipline was developed independently and then integrated to generate the final system. However, high quality designs cannot be achieved without simultaneously considering all the engineering disciplines. This mechatronic approach carries intrinsic complexity into system design process and numerous researches are on-going in order to find out optimal methods.
This article refines a SysML-based design process for the high level development of mechatronic systems, focusing on the integration of modelling and simulation as fundamental aspect for an integrated approach. How conceptual SysML diagrams may support the generation of simulation models is shown, along with a chain of simulations for an integrated design. The proposed approach is applied to a prototype case study designed from scratch, in order to be validated and to demonstrate its potentiality.

*Keywords:* SysML, Simulation, Modelling, MBSE, Mechatronic Systems.

## 1. INTRODUCTION

Mechatronic systems (MTSs) consist in the synergetic integration of mechanical engineering with electronic and intelligent computer control in the design and manufacturing of industrial products and processes, Harashima et al. (1996). The design of these systems requires a multidisciplinary approach through the integration of the single disciplines. Each discipline focuses on a particular aspect of the system and exploits different Domain Specific Models (DSMs) that should be merged and integrated. In order to cope with this situation, several tools implementing different capabilities are commercially available and start to have quite a large diffusion among industrial development communities (e.g. SysML [1], AutomationML [2], Lifecycle Modelling Language [3] etc.). However, the implementation of tools is not enough to develop performing systems, and optimal industrial processes are necessary.

The System Modelling Language (SysML) is a general-purpose graphical modelling language which is meant to become the "lingua franca" for MTSs modelling. Up to now, SysML has been mostly used as tool for reverse engineering processes; for example, Karban et al. (2011). Some tentatives were carried out in order to utilize it as supporting tool for the design process (e.g. Rosenberg and Mancarella (2010) for the development of an audio player, Weilkiens (2008) for an on-board computer for rental cars), but mostly for re-designing the behaviour of already existing systems concerning their interactions with the external world. In a previous work, Barbieri et al. (2014) introduce a SysML-based design methodology for MTSs. The integration of the involved domains is performed at two levels:

- Modelling (information) level; e.g. relating a requirement with the components which fulfil it.
- Simulation level; e.g. hardware in the loop simulation (HIL), in which the real controller is connected to a simulated physical model of the machine, Maclay (1997).

Purpose of this article is to link the two levels defined above. This link is pursued through the definition of a workflow. Then, the previous defined methodology is refined through the identification of a chain of simulations. These simulations can be performed in the different phases of the design process through the proposed workflow.

The paper is organized as follows: related works are described in section 2. Section 3 explains how to integrate modelling and simulation, and the chain of simulations inserted in the methodology. Section 4 resumes the design process and applies the proposed approach to a case study. Conclusions and future works are reported in section 5.

## 2. RELATED WORK

### 2.1 Model-Based Design

Model-Based Systems Engineering is the formalized application of modelling to support system requirements,

[1] SysML: http://www.omg.org/spec/SysML
[2] AutomationML: http://www.automationml.org/o.red.c/home.html
[3] Lifecycle Modelling Language: http://www.lifecyclemodeling.org/

火龙果・整理
uml.org.cn

design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases, INCOSE (2007). In the last years, this concept has been recognized as fundamental for a mechatronic design. In fact, there is the need to develop the designed system through an unique core model which works as the repository and dependency trace of the information of the different mechatronic domains. This is the idea of Model Based Design (MBD), El-Khoury et al. (2005). AutomationML and SysML may be used for this purpose.

AutomationML, introduced by Drath et al. (2008), reuses mature data formats to store information about topology (through CAEX), geometry and kinematics (through COLLADA), and logic (through PLCopen XML). The language is proper for connecting information contained in DSMs; e.g. the dimensions of components in CAD drawings. Whereas, SysML is used at higher level: for tracing and (above all) relating DSMs, but not necessarily the information contained into them. We think that the central model must be the most general as possible for containing information of all the involved domains. For this reason, we decided to utilize SysML for the application of the MBD approach. However, MBD is just a tool for realizing the integration among the different domains of MTSs and must be supported by an integrated design methodology.

### *2.2 Link conceptual and domain specific models*

MBD implies the exchange of information among the central SysML model and all the tools necessary for the design. There are two different ways for reaching this task:

- model transformation for generating DSMs from SysML models (either "manually" or automatically); for example SysML to Modelica, Paredis et al. (2010);
- connection with existing DSMs: utilization of the SysML model as a system representation and a repository of the system parameters and the DSMs. Different plugins are being born for implementing this connection (e.g. ParaMagic [4]).

Current tendency is to have a high level SysML model which contains an overview and traces dependencies among DSMs, and detailed SysML views which allow the one to one translation in DSMs, Shah et al. (2010). Every SysML view is modelled through a profile (Domain Specific Language, DSL) for mapping the semantic of the DSMs into SysML. Model Driven Engineering is also included in this category for the software viewpoint. Here, a higher layer of abstraction (e.g. SysML model, Chiron and Kouiss (2007)) is created and deployed in control code (e.g. Vogel-Heuser et al. (2005)).

However, these approaches lead to detailed SysML models with specialist views difficult to manage and to understand for people of other engineering domains. Moreover, unlimited extensions of the language would be generated, "destroying" the aim of a general purpose modelling language. For these reasons, we only create high level SysML models and we hyperlink them to DSMs.

[4] ParaMagic: http://www.intercax.com/products/paramagic

### *2.3 Conclusion*

In order to integrate mechatronic domains during design process, there is the need to identify a method for integrating SysML modelling and simulations performed in DSMs. This should be reached without extending the SysML language with DSLs. A proposal is illustrated in Sec. 3.

## 3. INTEGRATION OF MODELLING AND SIMULATION

### *3.1 The proposed workflow*

Following the criteria defined in Sec. 2, we propose the workflow shown in Fig. 1, which consists of:

- Conceptual model: SysML diagrams for rationalizing what will be implemented in the simulation tool. Which diagrams to utilize depends on the performed simulation. For example, for dynamics simulations, an internal block diagram (ibd) can be used for representing the structure, and a state machine diagram (stm) for the behaviour.
- DSM implementation: implementation (and not one to one translation) of the conceptual SysML diagrams in a simulation environment and model execution. Then, SysML model is hyperlinked to the generated DSM in order to trace its (file) path. A possible implementation at the example of a dynamics simulation performed in MATLAB [5] is described next:
  (1) Each block of the ibd becomes a sub-system in the Simulink environment. Each block implements the dynamics equations of the represented object.
  (2) Parameters are assigned to the equations (e.g. dimensions, friction coefficients etc.).
  (3) All the sub-systems are connected as indicated in the ibd for generating the system model.
  (4) The behaviour represented in the stm is implemented in Stateflow.
  (5) An integrated simulation is performed linking the Simulink and Stateflow models.

  Comparing the simulation implementation described above and a standard modelling process, it can be noticed that the validation of the model is missing. Validation is generally performed through the comparison of simulation results with experiments. However, a physical prototype is usually not available in all the phases of the design process. In fact, different alternatives for fulfilling a functionality can be developed and the generation of a physical prototype for each alternative might result in a long and expensive activity. For this reason, simulation is performed for comparing alternatives modelled assigning same values to the common parameters. In this way, alternatives are evaluated on the basis of their plausible behaviour. When the real physical behaviour is identified, one can perform again the comparison in order to validate the results.
- Failure-modes analysis: evaluation of all the possible failures and determination of new functions for making the system work correctly; e.g. definition of functions for fault detection and fault management.

[5] MATLAB: MathWorks www.mathworks.com

火龙果・整理
uml.org.cn

- Refinement: refinement of the requirements and of the conceptual simulation models; e.g. the need of a strategy for managing a certain fault is found out.
- Comparison matrix: if different alternatives must be evaluated, a comparison is performed on the basis of the trade-off criteria and weights defined in the stakeholder requirements.

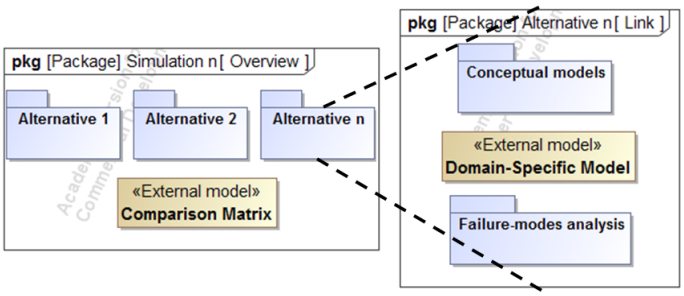


Fig. 1. SysML template for the integration of modelling and simulation at the example of the simulation type "n".

### *3.2 Benefits of the approach*

In short, the approach consists in the creation of conceptual SysML diagrams, in their "manual" implementation in simulation environments, and in the hyperlink of the SysML model to the generated DSMs. This brings the following benefits:

- Simulation rationalization: SysML language provides a tool for rationalizing the simulation models without starting the simulation activity from the blank canvas; for example, boundaries of components can be defined along with interfaces, simulation objectives etc. SysML model contains high level information without all the details which would have required the implementation of DSLs and transformations from SysML to DSMs (and vice-versa) for keeping consistency.
- Comprehension: simulation is rationalized in a general purpose modelling language and not in the language of the DSM. People, not expert in a certain simulation environment, can understand high level information concerning that environment.
- Traceability: SysML model traces the file path of DSMs and indicates their role in the design process; for example, a certain DSM is used for verifying a requirement.

### *3.3 The proposed chain of simulations*

Different granularities of simulations can be performed for designing MTSs, Lasa et al. (1999). A list is reported below sorted on the basis of the required level of detail and computational efforts, from the coarsest to the finest. Simulation categories are illustrated with typical use cases:

- Discrete event simulation (DES): mainly used for the simulation of lines and plants where only functional behaviour is important.
- Virtual commissioning simulation (VC): for virtual debug of control logic (usually through HIL), but also for operator training, conceptual design and communication, and control logic definition; usually performed through kinematics or simplified (i.e. differential equations solved with high tolerances) dynamics simulations.
- Dynamics simulation (DS) and multibody analysis: for the study of the physical behaviour of interconnected rigid or flexible bodies; usually utilized for dimensioning type, size and tuning of actuators.
- Finite element method (FEM): used for problems where geometry plays an important role as structural analysis, magnetic fields, turbulent flow etc.

DES, VC and DS allows the simulation of both the physics and control logic of the designed system. For this reason, they constitute a fundamental step for an integrated design and will be inserted in the proposed approach for:

- DES: for deriving functional attributes from target requirements; e.g. lot size, cycle time etc.
- VC: for comparing different physical and control logic alternatives and for deriving kinematics parameters; e.g cam profile of actuators.
- DS: for deriving necessary physical parameters which will guide the choice of commercial components; e.g. required torque of electrical motors.

## 4. CASE STUDY

### *4.1 Introduction*

In Barbieri et al. (2014), we propose an integrated design through the application of the W model. Fig. 2 illustrates the different phases of the process. Every phase becomes a package in SysML and contains SysML diagrams in order to guide designers during the development process and to trace information.

Next, the methodology is resumed.

(1) *System Context*: definition of the context in which the system will operate and of the high level requirements.
(2) *System Analysis*: definition of the technical requirements, the functions the system must perform and system subdivision in modules.
(3) *Specific Solutions and Dependency Analysis*: break down of the system requirements in module requirements and definition of feasible means for fulfilling them.
(4) *Virtual System Integration*: combination of the identified means for generating different alternatives of the system, and system simulation in order to compare different physical and control logic solutions, refine the requirements and identify necessary physical parameters. DES, VC and DS are performed through the workflow identified in Sec. 3.1.
(5) *Domain Specific Design*: detailed development of the modules and selection of commercial components through the application of domain specific methods and tools; e.g. FEM simulation for dimensioning components during operational and static conditions.
(6) *System Integration*: integration of the designed modules for generating the final system and Verification and Validation (V&V) through simulations and ex-

火龙果・整理
uml.org.cn

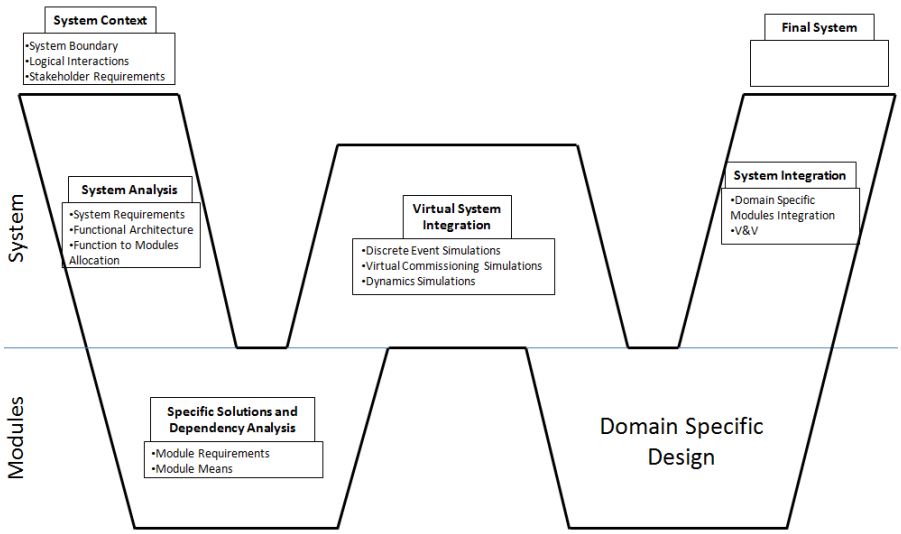


Fig. 2. The proposed W model

perimental tests. DS, HIL and DES can be utilized in this phase.

(7) *Final System*: folder which contains the information useful for the reuse into other projects; for example requirements, interfaces, DSMs information etc.

After having recall the design methodology, its application on a case study is shown. The study was performed on a prototype MTS developed from scratch: a transport system for sorting packages. The main purpose was to test the whole methodology, and the benefits of using the proposed workflow and the identified chain of simulations. Whereas, we were not interested on designing a real working system. For this reason, we just performed DES and VC simulations, while we skipped DSs and the choice of commercial components. All the SysML diagrams were implemented through MagicDraw [6]. Next, the study is shown with a deep explanation of the parts relative to the simulation workflow.

### *4.2 System context*

Project mission was to create a system able to sort inlet packages in group of either three or four on the basis of the operator choice. Starting from that, the context in which the system operates was defined, along with the logical interaction (i.e. static interface and communication protocol with the external world), and requirements from possible stakeholders.

### *4.3 System analysis*

Stakeholder requirements were broken down in system requirements. The system functional architecture was then individuated and the system was decomposed in three modules, assigning them the functions they must fulfil:

- Module 1 $\longrightarrow$ "Collect packages from the upstream conveyor", "Provide packages to the downstream conveyor" and "Transport packages";
- Module 2 $\longrightarrow$ "Sort packages";
- Module 3 $\longrightarrow$ "Discharge unsorted packages".

[6] MagicDraw: No Magic Inc. http://www.nomagic.com/

### *4.4 Specific solution and dependency analysis*

System requirements were broken down in module requirements and feasible means were associated to the modules. These had to implement the functions individuated in the system functional analysis (e.g. a mechanical feeder is able to sort packages pushing them into the downstream conveyor).

- Module 1 $\longrightarrow$ Conveyor;
- Module 2 $\longrightarrow$ Obstruction (blocks packages and releases them when the proper number is accumulated), Mechanical feeder, Robot (picks and places packages in the downstream conveyor), Brake (as the obstruction solution but with the difference that packages are slowed down and not stopped) and Divider+Obstructions (longitudinal bar which places packages transported from a conveyor in different positions along an obstruction; obstruction is removed when the proper number is accumulated);
- Module 3 $\longrightarrow$ Pressured air.

### *4.5 Virtual system integration*

The identified means were integrated, generating different alternatives of the system. Then, simulations were performed in order to determine the best solution, and to define the control logic and kinematics parameters. We utilized FlexSim [7] for DES and IndustrialPhysics [8] for VC. The proposed workflow was applied for both the simulations:

- Conceptual simulation model: DES: an activity diagram was defined for each alternative. Each diagram displays the performed functions in a logical order. Then, the activities were divided in two categories: the ones which are automatically executed by means (e.g. a wall automatically stops packages which collide) and the ones which needs actuators, sensors and a control logic to be performed. These last activities were refined through state machine diagrams. Fig. 3

[7] FlexSim: http://www.flexsim.com
[8] IndustrialPhysics: http://www.machineering.de

火龙果・整理
uml.org.cn

illustrates the conceptual diagrams for the robot alternative. VC: internal block diagram for representing the system structure and the exchanged items, and state machine diagram for the system behaviour.

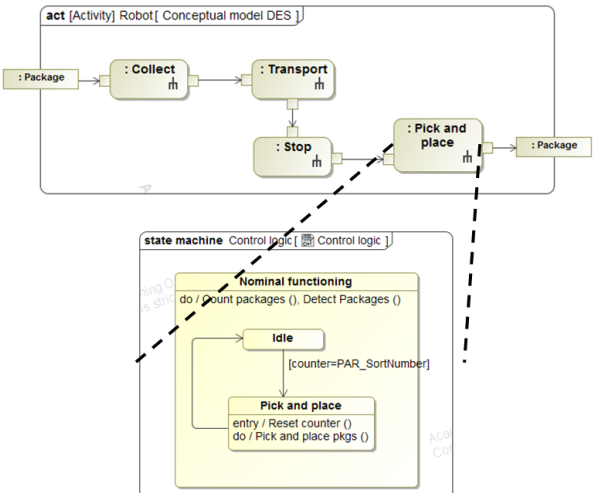


Fig. 3. DES rationalization: activity diagram of the robot alternative, and state machine diagram of the pick and place activity. At this level of the design, just the nominal functioning was considered because the most relevant for the trade-off analysis.

- Simulation implementation: implementation of the conceptual models in the simulation environment. DES: activities are implemented by FlexSim objects and the control logic is inserted through internal scripts, Fig. 4. VC: each SysML block becomes an object in IndustrialPhysics and the control logic is inserted through internal scripts.

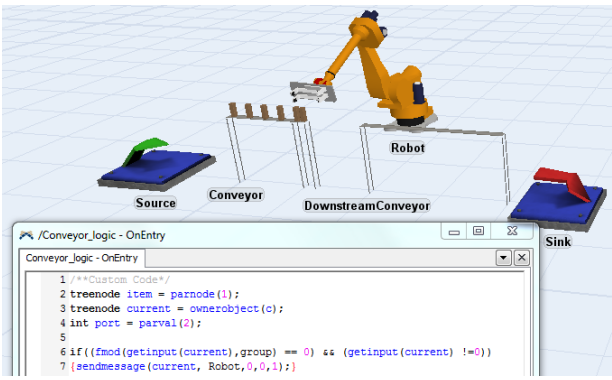


Fig. 4. DES of the robot alternative. The activities defined in SysML are implemented in FlexSim: collect packages → connection between the source and the conveyor; transport → conveyor; stop → opening and closing of the conveyor output port; pick and place → robot. The control logic is implemented in a script.

- Failure-modes analysis: different failures were found out through the simulated models. These were documented in a table, which contains the functions related to that failure and the derived requirements for managing it. An example is shown in Fig. 5.

  For the failure-modes analysis through different types of simulations is important to remind that: DES is a functional simulation based on statistical distributions and time gains, while VC a physical simulation. For this reason, failures can be identified in VC, whereas can only be inserted through statistical distributions in DES. For example, the fall down of packages due to the collision with a pneumatic actuator which has an incorrect motion profile, can be identified just through VC simulation. Consequently, in the context of failures managing, DES was utilized for determining timing aspects (e.g. available time) and control logic, while VC for identifying possible failures.

| Name | Function | Derived |
|---|---|---|
| 1. Packages fall down from the conveyor | Transport | 349 Guide packages |
| 2. Packages are blocked by the guides | Transport | 348 Guide gap |
| 3. Bad detection for not vertical packages | Detect fall packages | 368 ToleranceInDetectingFallPackages |
| 4. Discharge sorted packages | Discharge unsorted | 347 Wait before discharge |

Fig. 5. Failure analysis of the obstruction solution during VC simulation.

- Refinement: refinement of the conceptual models and of the requirements with the results of the simulation and the failure-mode analysis. For example, after VC, requirements and control logic for managing fall and attached packages were inserted.
- Comparison matrix: comparison of the different alternatives through the trade-off requirements, attributing different weights on the basis of the stakeholders objective hierarchy. An example is shown in Fig. 6.

After DES and VC simulations, the obstruction solution resulted as the most promising one. For this reason, the other solutions were discharged and the obstruction was deeper designed. As an iterative process, if in the next phases this solution was not acceptable, we would come back to this phase and choose the second one. In the same way, more than one alternative might be designed and another trade-off executed subsequently.

| | Axis nr (3) | Sensors nr | ... | Configuration change (2) | Total |
|---|---|---|---|---|---|
| **Obstruction** | 3 | 3 | | 3 | **27** |
| **Trolley** | 1 | 3 | | 4 | **22** |
| **Mechanical feeder** | 3 | 3 | | 2 | **26** |
| **Divider** | 1 | 1 | | 4 | **20** |
| **Robot** | 2 | 3 | | 2 | **25** |

Fig. 6. Excel comparison matrix after the DES of the different alternatives. A grade from one to three was assigned for every criteria and multiplied for the weight inside the brackets.

### *4.6 Domain specific design*

As indicated before, we skipped the choice of commercial components and the domain specific design phase. We just implemented the control logic defined in the system integration phase in PLC code. The design pattern, proposed by Fantuzzi et al. (2011), was utilized. It consists in a centralized control in which every object becomes a modular PLC program which supervises and synchronizes its child objects (i.e. other PLC programs). The process is shown in Fig. 7 at the example of the conveyor module.

### *4.7 System integration*

Control code was debugged through HIL. The real controller (Rockwell PLC) was connected to a virtual model

火龙果・整理
uml.org.cn

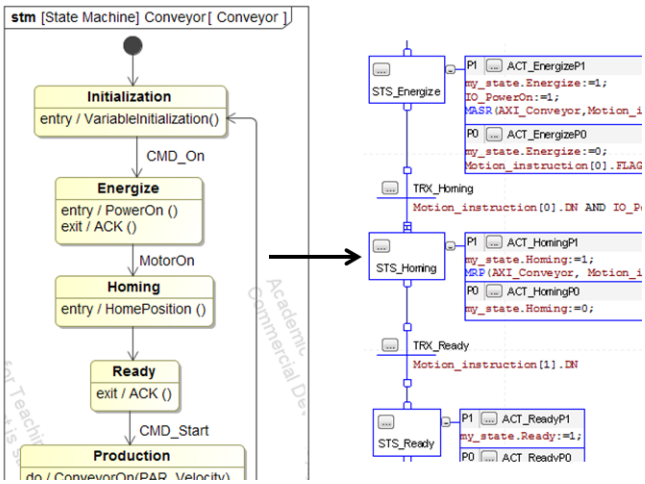


Fig. 7. Implementation of the conveyor behaviour defined in the virtual system integration phase (left side) in IEC 61131-3 Sequential Function Chart (right side).

of the machine (IndustrialPhysics model). Eventually, the "satisfy by" and "verify by" compartment of the requirement tables were filled.

### *4.8 Final system*

Requirements, interfaces and DSMs information were inserted in this folder. However, guidelines and a template for the representation of the final folder information will be matter of future investigation. The idea is to structure the folder in a way that, copying it (or part of it) into another project, you can have a re-usable MTS (or sub-system).

## 5. CONCLUSION AND FUTURE WORKS

This work has refined a MBD methodology defining a workflow for integrating modelling and simulation, and identifying which simulations may be performed for an integrated design. The approach was applied in the design of a prototype MTS for sorting packages. The study started from the stakeholder requirements and, through the implementation of the design methodology and the defined chain of simulations, we were able to virtually design (at high level) the whole system.

Next, future works are illustrated. In this paper, we showed how the domains involved in MTSs design can be integrated through a chain of DES, VC and DS simulations. However, the borders among DES, VC and DS are not always clear. For example, kinematics parameters might be identified both with DES and VC. A deeper investigation will be conducted for defining application use cases and borders of each simulation.

Moreover, there are many other types of available simulations; see Ptolemaeus (2014) for an overview. A study will be conducted in order to:

- define which simulation types allow the integrated design of MTSs;
- define when these simulations should be performed in the proposed design process.

Eventually, requirements, interfaces, and DSMs information are inserted in the final system folder. However, a definition of the information that this folder should contain and a template for its representation has not been investigated yet.

## REFERENCES

Barbieri, G., Fantuzzi, C., and Borsari, R. (2014). A Model-Based Design Methodology for the Development of Mechatronic Systems. *Mechatronics Journal*.

Chiron, F. and Kouiss, K. (2007). "Design of IEC 61131-3 function blocks using SysML". In *Mediterranean Conference on Control Automation, MED*.

Drath, R., Luder, A., Peschke, J., and Hundt, L. (2008). AutomationML - the glue for seamless automation engineering. In *Emerging Technologies and Factory Automation (ETFA)*. Hamburg, Germany.

El-Khoury, J., Redell, O., and Törngren, M. (2005). A Tool Integration Platform for Multi-Disciplinary Development. In *Software Engineering and Advanced Applications*.

Fantuzzi, C., Bonfé, M., Fanfoni, F., and Secchi, C. (2011). A Design Pattern for translating UML software models into IEC 61131-3 Programming Languages. In *18th World Congress of International Federation of Automation Control (IFAC)*.

Harashima, F., Tomizuka, M., and Fukuda, T. (1996). *"Mechatronics: What is it, Why and How?"*. IEEE/ASME Transactions on Mechatronics 1.

INCOSE (2007). Systems Engineering Vision 2020. Technical report, Intern. Council on Systems Engineering.

Karban, R., Weilkiens, T., Hauber, R., Zamparelli, M., Diekmann, R., and Hein, A. (2011). *CookBook for MBSE with SysML*. SE2 Challenge Team.

Lasa, M., Heinkel, H.M., Moser, E., and Rothfuss, R. (1999). Modeling Mechatronic Systems at Different Levels of Abstraction. In *Magdeburger Maschinentage*.

Maclay, D. (1997). Simulation gets into the loop. *IEE Review*.

Paredis, C.J., Bernard, Y., Burkhart, R.M., de Koning, H.P., Friedenthal, S., Fritzson, P., Rouquette, N.F., and Schamai, W. (2010). An Overview of the SysML-Modelica Transformation Specification. In *INCOSE International Symposium*.

Ptolemaeus, C. (ed.) (2014). *System Design, Modeling, and Simulation using Ptolemy II*. Ptolemy.org. `http://ptolemy.org/books/Systems`.

Rosenberg, D. and Mancarella, S. (2010). *Embedded Systems Development using SysML*. OMG: Systems Modeling Language.

Shah, A.A., Kerzhner, A.A., Schaefer, D., and Paredis, C.J.J. (2010). Multi-View Modeling to Support Embedded Systems Engineering in SysML. In *Graph Transformations and Model-Driven Engineering*.

Vogel-Heuser, B., Witsch, D., and Katzke, U. (2005). Automatic Code Generation from a UML model to IEC 61131-3 and System Configuration Tools. In *International Conference on Control and Automation*.

Weilkiens, T. (2008). *Systems Engineering with SysML/UML: Modeling, Analysis, Design*. Morgan Kaufmann Publishers Inc.