|  |  |
| --- | --- |
| Dx.y | Deliverable Title |
| Access[[1]](#footnote-1): | **PU** |
| Type[[2]](#footnote-2): | **Report** |
| Version: | **1.0** |
| Due Dates[[3]](#footnote-3): | **M12, M24, M36** |
| Environment for model-based rigorous adaptive co-design and operation of CPS | |
| **Executive summary[[4]](#footnote-4):** | |
| This deliverable updates the state of the art presented in the FPP. | |

**Deliverable Contributors:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Name | Organisation | Primary role in project | Main Author(s)[[5]](#footnote-5) |
| Deliverable Leader[[6]](#footnote-6) | Lena Buffoni | LiU | Project coordinator | X |
| Contributing Author(s)[[7]](#footnote-7) |  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
| Internal Reviewer(s)[[8]](#footnote-8) | Adrian Pop | RISE | WP3 Leader |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Document History:**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Reason for Change | Status[[9]](#footnote-9) |
| 1.0 | 15/01/2021 | First Issue | Released |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Contents**

[Abbreviations 3](#_Toc61607487)

[1 Introduction 4](#_Toc61607488)

[1.1 Formal requirements modelling and simulation 4](#_Toc61607489)

[1.2 Multi-facet modelling and simulation (including multi-mode reconfigurable M&S) 6](#_Toc61607490)

[1.3 Safety and dependability modelling and simulation 7](#_Toc61607491)

[1.4 Large scale modelling and simulation 7](#_Toc61607492)

[1.5 Collaborative modelling and simulation frameworks 8](#_Toc61607493)

[References 9](#_Toc61607494)

# Abbreviations

List of abbreviations/acronyms used in document:

**Abbreviation Definition**

FMI Functional Mock-up Interface

FMU Functional Mock-up Unit

M&S Modelling and Simulation

N/A Not Applicable

SotA State of the Art

TBD To Be Defined

# Introduction

Methodologies for the engineering of CPS that aim at covering the value chain (cf. Figure 1) should deal with the following issues:

1. How to express multi-faceted requirements consistently while keeping independent from design solutions and verify their compliance vs. the assumptions on the system and its environment?
2. How to derive design from requirements and automate the verification of design vs. requirements?
3. How to explicitly take real-time constraints into account?
4. How to deal in practice with the fact that physical systems are subject to uncertainties that prevent requirements to be always satisfied?
5. How to optimize the system w.r.t. to the multi-faceted constraints?

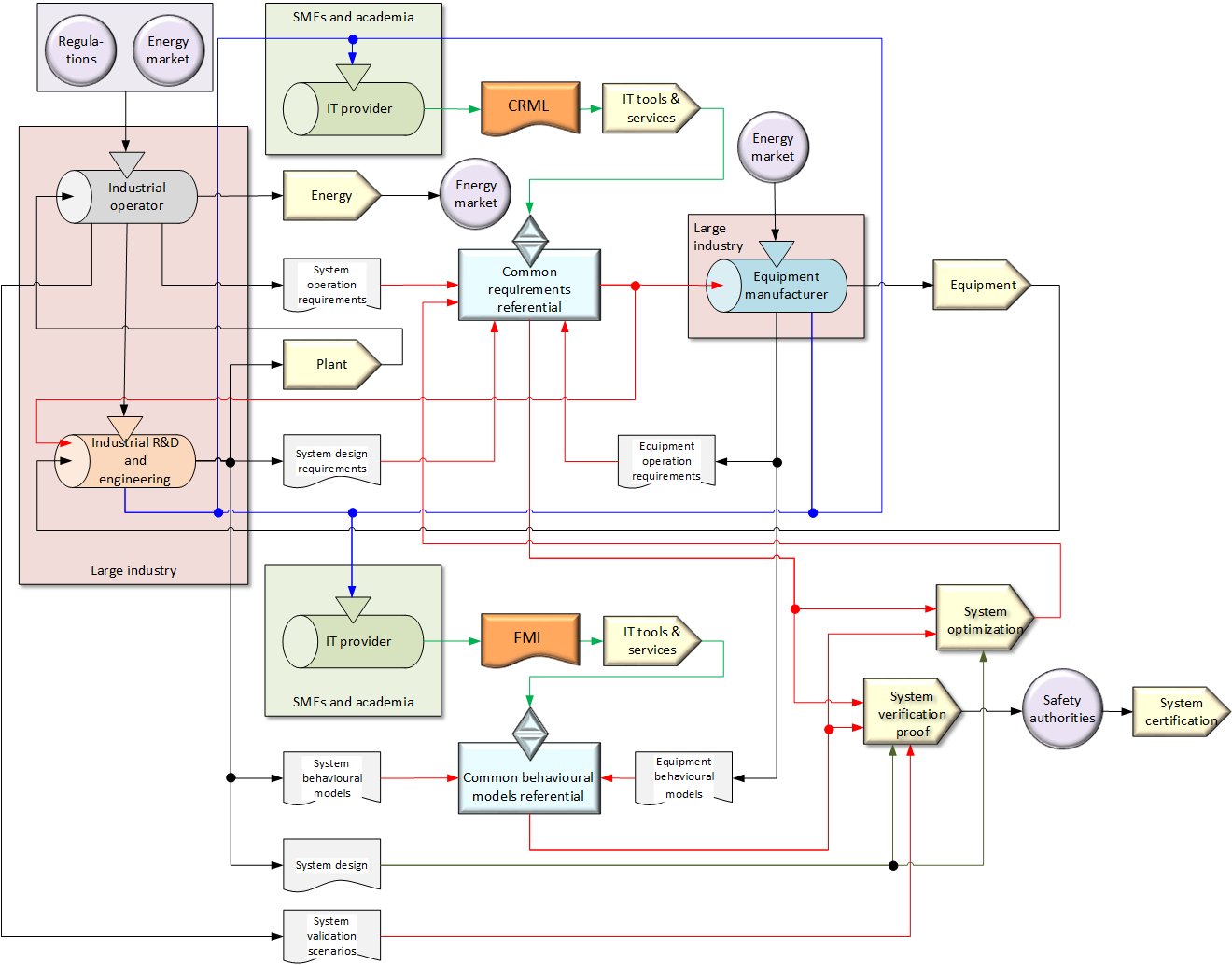


Figure 1: value chain

The current state-of-the-art regarding these issues tends to consider them separately. Consequently, there is no integrated tool able to deal with all of them in a consistent way. The main gaps concern the links between logical design, physical design and dependability analysis as they currently involve completely different methods and tools: logical design uses methods such as UML/SysML based on first order logic that originates from the software industry, physical design uses tools such as Modelica tools, Matlab, Simcenter Amesim, etc. that deal with physical laws in the form of DAEs (or block diagrams) and dependability analysis uses probabilistic methods. Several attempts that are not widely used in practice were made to bridge the gaps, the most promising being those integrating the SysML standard with the Modelica standard.

Besides, there is no convincing solution to express the requirements independently from design solutions in a formal way, so this is currently done by expressing requirements in natural language with tools such as Rational DOORS, Polarion, etc. This comes from the fact that formal requirements modelling methods tend to bear on abstractions of the system in the form of state machines, which already express a solution.

In the sequel, more detail is given on the state-of-the-art concerning these issues.

## Formal requirements modelling and simulation

There are different types of formal requirements modelling languages, such as:

* Temporal logic for model checking such as LTL and CTL (Baier, 2008);
* Timed and hybrid automata;
* Finite state machines such as the state machines diagrams of UML or SysML.

All these have the following main limitation with respect to CPS: they consider systems as finite state machines. For instance, with LTL, one can prove that a system will always or eventually pass through a given state. Timed automata can handle real-time, but only when the states are known in advance. This corresponds to an idealistic view of the system that is not for instance subject to wear or external aggression. Hence they do not consider situations where existing states are subject to gradual drift due to wear, or new states appear due unexpected events. In other words, CPS contain finite-state machines, but they cannot as a whole be considered as finite-state machines.

Other limitations are:

* Lack of object-orientation: temporal constraints cannot be (easily) associated to the system architecture (i.e. its decomposition into subsystems and components).
* Difficult mathematical syntax: although mathematical syntax (and semantics) is necessary to perform formal proofs (model checking) or even model simulation, it is difficult to use on a day-to-day basis for the whole system.
* Difficulty to handle real-time constraints.

Various attempts have been made to alleviate those limitations by extending OCL to temporal constraints, but none of them are used convincingly in practice (Kanso and Taha, 2013).

Therefore, formal requirements languages are usually only used for small (sub)systems with critical safety issues, and they cover only the very early phases of system design when the logics of the system is investigated.

Modelica is a formal language that comes with a convenient graphical interface fit for the description of the physical real-time behaviour of CPS. However, Modelica cannot express constraints on a system when its architecture is not known. Therefore, Modelica cannot be used for the early system design phases. Graph-based design languages however with their capability to explicitly modify product topology and parametrics are on the one hand partially able to fill this gap, but need to be extended on the other hand by more powerful formal methods for requirements processing, tracing and consistency checking.

Therefore, connecting formal requirements modelling languages such as the ones mentioned above directly to Modelica does not solve the general problem of having a model-based methodology that covers the whole engineering lifecycle for CPS, as such kind of solutions is only valid if the system is considered as a state-machine and for the engineering phases past the detailed design phase (which is somewhat contradictory).

The specifications and an architecture for a new modelling language named FORM-L for the capture of requirements and assumptions adapted to CPS have been produced in the scope of the MODRIO project (Bouskela et al., 2017). First ideas for a new temporal language named ETL (Extended Temporal Language) aimed at the simulation of the temporal aspects of FORM-L have been expressed, and an implementation of ETL into a Modelica library called ReqSysPro has been produced (Bouskela and Jardin, 2018). They will be used as starting point for the development of the new language called CRML and its associated framework (cf. §2.3.3).

Contracts are an elegant and powerful concept for capturing system or program requirements. The approach was first developed and promoted in software engineering. This has been popularized with the Eiffel object-oriented programming language, designed by Bertrand Meyer (Meyer, 2009) reusing earlier ideas from Floyd-Hoare logic (Floyd, 1967, Hoare, 1969). Floyd-Hoare logic assigns meaning to sequential imperative programs in the form of triples of assertions consisting of a precondition on program states and inputs, a command, and a postcondition on program states and outputs. So far contracts consisting of pre/postconditions naturally fit imperative sequential programming. In situations where programs may operate concurrently, interference on shared variables can occur. Rely/Guarantee rules (Jones, 1983) were thus added to interface contracts. Rely conditions state assumptions about any interference on shared variables during the execution of operations by the system's environment. Guarantee conditions state obligations of the operation regarding shared variables. Floyd-Hoare or Separation logics are however unsuitable for reasoning about CPS: they cannot cope with continuous-time dynamics, and they cannot be easily adapted to capture the behaviour of components in constant interaction with their environments through input and output streams, as this is typically the case in CPS.

Assume/Guarantee contracts (Benveniste, 2015) are best suited to formalize CPS requirements and support a Component-based Design approach for CPS, even during early stages of design. When the design of a component is undertaken, implicit, and often hidden assumptions regarding the environment in which the component will evolve are typically invoked by the designer. Assume/Guarantee contracts allow to pinpoint responsibilities and make hidden assumptions explicit.

## Multi-facet modelling and simulation (including multi-mode reconfigurable M&S)

Multi-facet modelling is the modelling of the different aspects of the same system corresponding to the different disciplines (cf. Figure 3). Facets are statically linked together using bindings. Using bindings, a variable of interest from a given discipline (e.g. economics) can be used in another one (e.g. physics). Figure 2 shows the bindings between the requirements model and the other disciplines (the yellow arrows), but other kinds of bindings are possible (e.g. between economics and physics).

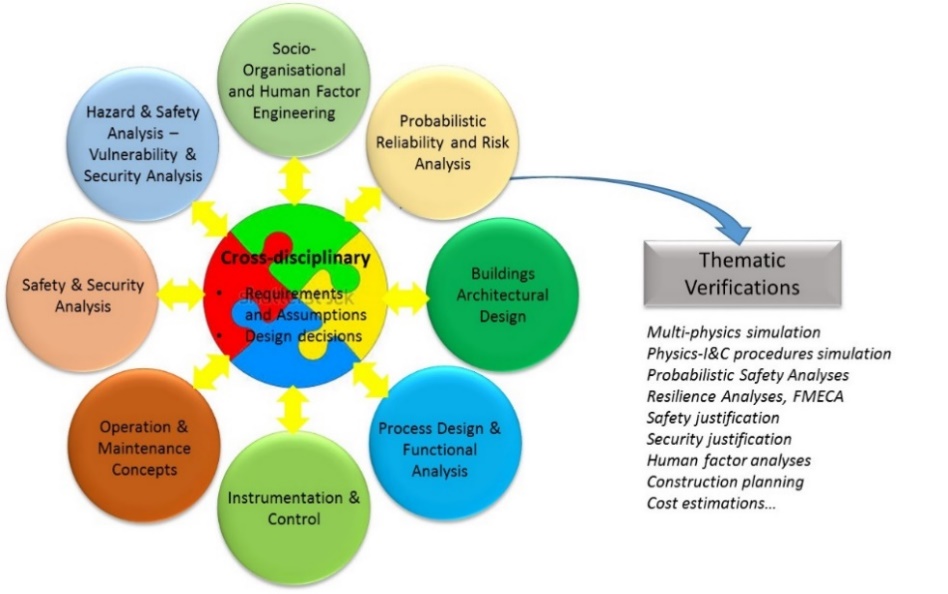


Figure 2 engineering disciplines sharing a common requirement modelling view

The models of the different facets of a system are now developed in isolation. Simulating them together can be achieved using co-simulation, e.g. using the FMI or other kind of co-simulation protocol, like DCP (formerly known as ACI), which was developed within the ACOSAR project and is continued as Modelica Association Project. However, this remains a difficult task because when models are developed by different teams, they are not conceived to be co-simulated, so they need to be adapted to that purpose, which is a costly and discouraging endeavour. This is why EMBrACE will further develop the mechanism of bindings, whose purpose is to be able to exchange variable of interests between models without having to modify the models (as long as the variables of interest are present in the models and can be exported). An architecture for supporting such bindings has been proposed in the framework of the MODRIO project (Bouskela et al., 2017).

Multi-mode modelling is the modelling of the different behavioural modes (e.g. normal, dysfunctional…) corresponding to the same discipline, whose switching (e.g. from normal to dysfunctional) occurs dynamically at run-time when an event occurs (e.g. component breakage). Reconfigurable modelling is the modelling of systems with appearing or disappearing subsystems (such as plants being started or stopped on a power grid, or cars entering or leaving a district).

Multi-mode reconfigurable M&S is necessary to be able to explore the consequences of an initiating event on the system that may lead to mode switching. It is thus necessary for designing and verifying fault-resilient systems against requirements. Models with multiple modes exhibit time-varying structure: when switching modes, state variables may appear or disappear dynamically. There is a well-known approach in automata theory called hybrid automata that is able to handle multi-mode systems if the modes are described by ordinary differential equations (ODEs). However, for CPS this approach is not feasible because the physical equations in the form of implicit DAEs must be converted into explicit ODE form, and the number of needed states to do so can grow very rapidly.

Therefore the simulation of mode switching in CPS is a fundamental difficulty that has been recently successfully addressed within the MODRIO project by designing and implementing the prototype of a new Modelica hybrid-state machine in Dymola (Elmqvist et al., 2014). However, this prototype is still not general enough to address all kinds of systems. In particular, it does not address reconfigurable systems.

Graph-based Design Languages in UML are good at modelling the evolving system architecture through a set of model transformations. Due to the graph-based representation, the changes in system architecture are explicit (i.e. through the graph manipulations) and can easily be translated into other digital representations (e.g. through the transformation of UML into arbitrary other formal information representations). Design Languages are therefore a potential means to build a framework able to interconnect various other modelling languages such as Modelica or CRML.

## Safety and dependability modelling and simulation

Standard dependability models (such as fault trees, Bayesian networks, Petri nets, etc.) are always dedicated to a single system, not to say a single function or point of interest of a system. Their construction is difficult and cumbersome, and they must be checked and updated whenever the system design or operation changes.

In order to avoid this inconvenience, object oriented domain specific languages (DSLs) for dependability modelling have been created. Some of them are defined as UML profiles. Some others are textual descriptions with a precisely defined syntax and semantics (Figaro, AltaRica, AADL, O3PRMhttps://o3prm.gitlab.io/…). Thanks to these DSLs, it is possible to generate fault trees, Bayesian networks and stochastic simulation models, having only one model to maintain for a given system. Moreover the structure of the dependability model can be very close to the structure of the system, which makes it easier to establish a binding between the two models. The feasibility of a binding between a Modelica model and a Figaro model has been demonstrated in the MODRIO project. However, this approach can be inapplicable in a few situations, as both the Modelica model and the dependability model may, in particular for reasons of efficiency in the calculations, have structures that do not match the physical design of the system. On the other hand, UML based tools aim at producing dependability models as "enriched" versions of the UML design models. This approach is not satisfactory because the UML design models are usually built in a "loose" way, as they are essentially a communication means between humans, and therefore they cannot be a valid support for automatic processing.

Given the limits of existing approaches, in EMBrACE the aim will be to derive the dependability model - as much as possible automatically - from the requirements and the description of a more or less detailed design in the CRML language.

## Large scale modelling and simulation

The design and safe operation of modern large-scale CPS require the ability to model and simulate them efficiently. The Modelica language is optimally suited for the modelling task, thanks to the high-level declarative modelling approach and to the powerful object-oriented features such as inheritance and replaceable objects. On the other hand, until recently the development of Modelica tools has been focused on the modelling of moderate-sized models, optimizing the simulation code as much as possible by means of structural analysis and symbolic processing of the system of equations.

Large system models are usually characterized by a high degree of sparsity, since each component interacts only with a few neighbours, so that each differential-algebraic equation in the model only depends on a handful of variables. The availability of reliable open-source sparse solvers and of cheap computing power and memory even on low-end workstations opens up the possibility of tackling much larger system models, featuring hundreds of thousands or possibly millions of equations, exploiting the sparsity of such models for their solution. In particular, the interest in the use of Modelica for the modelling and simulation of national- and continental-sized power generation and transmission systems recently motivated a first exploratory effort in this direction, using OpenModelica as a development platform. The methods implemented for the power system studies also allowed to efficiently simulate the cooling blanket of the future DEMO nuclear fusion reactor, which requires the modelling of thousands of individual heat-exchanging pipes.

Modelling and Simulation of large multi-mode systems requires parallel algorithms. For example, the German Partners FH Bielefeld currently develop and improve such methods within the HPC-project PARADOM (Parallel Automatic Differentiation in OpenModelica) which is funded by the German government. These methods have been available at the start of EMBrACE. In PARADOM the Technical University Dresden in particular supports the development of efficient exploitation of Cache-hierarchies.

## Collaborative modelling and simulation frameworks

Engineering energy or transport systems involve concurrent design cycles with various teams working on a wide range of fields, and relying on modelling and simulation. It is widely recognized that improved collaboration and model exchange between all stakeholders is a key enabler for cost and time-to-market reduction.

On the other hand, web and cloud-based solutions are increasingly being used in several areas of business intelligence. Their strength is to allow different actors, located everywhere, to build, access and share common views that are related to common project they are working on.

However, to the best of our knowledge, no such tools exist to support the articulation between systems engineering with modelling and simulation. The closest solutions are purely focused on modelling and simulation. They comprise the [InsightMaker](https://insightmaker.com/) framework for collaborative modelling and simulation, based on the systems dynamics formalism and [Aperion](https://www.xogeny.com/products/)by the Xogeny company that allows to build web applications around FMI simulations. Both are developed in the USA. A new web-based collaboration tool targeting modelling and simulation has been developed by Modelon (<https://www.modelon.com/modelon-impact/>). Alternative open-source solutions are developed by RISE in project HUBCAP.

# References

Baier C. and Katoen J.-P. (2008), “Principles of Model Checking,” The MIT Press.

Benveniste A. et al. (2015), “Contracts for Systems Design: Theory, Methodology and Application Cases”, https://hal.inria.fr/hal-01178467, <https://hal.inria.fr/hal-01178469>.

Bouissou M., Buffoni L. and Thiele B. (2016), “From design to dependability: a bridge between physical simulation and risk analysis,” lambda-mu 20, St Malo.

Bouskela D., Nguyen T. and Jardin A. (2017), “Toward a Rigorous Approach for Verifying Cyber-Physical Systems Against Requirements,” Canadian J. of Electrical and Computer Engineering, Vol. 40-2, pp. 66-73.

Bouskela D. and Jardin (2018), “ETL: A New Temporal Language for the Verification of Cyber-Physical Systems,” 2018 Annual IEEE International Systems Conference (SysCon).

Elmqvist H., Mattsson S.E. and Otter M. (2014), “Modelica extensions for multi-mode DAE systems,” 10th Int. Modelica Conf. proceedings.

Floyd R. (1967), “Assigning Meaning to Programs”, Proc. Symp. Appl. Math, vol. 19.

Hoare T. (1969), “An Axiomatic Basis for Computer Programming”, Comm. ACM, 12(10).

Jones C. (1983), “Specification and Design of (Parallel) Programs”, IFIP Congress.

Kanso B. and Taha S. (2013), “Temporal Constraint Support for OCL,” in: Czarnecki K., Hedin G. (eds) Software Language Engineering. SLE 2012. Lecture Notes in Computer Science, vol. 7745. Springer, Berlin, Heidelberg.

Mattsson S.E. (2015), “Multi-Mode DAE Systems with Varying Index,” 11th Int. Modelica Conf. Proceedings.

Meyer B. (2009), “Touch of Class: Learning to Program Well Using Object Technology and Design by Contract”, Springer.

VDI/VDE (2015), “Gesellschaft Mess- und Automatisierungstechnik, “Reference Architecture Model Industrie 4.0 (RAMI4.0)”.

Walter B., Hammes J., Piechotta M. and Rudolph S. (2017), "A Formalization Method to Process Structured Natural Language to Logic Expressions to Detect Redundant Specification and Test Statements." In Requirements Engineering Conference (RE), 2017 IEEE 25th International, pp. 263-272.

Walter B., Kaiser D. and Rudolph S. (2018), “Machine-executable Model-based Systems Engineering with Graph-based Design Languages.” Poster, Proceedings CSD&M 2018, Paris, France.

1. Access classification as per definitions in PCA; PU = Public, CO = Confidential. Access classification per deliverable stated in FPP. [↑](#footnote-ref-1)
2. Deliverable type according to FPP, note that all non-report deliverables must be accompanied by a deliverable report. [↑](#footnote-ref-2)
3. Due month(s) according to FPP. [↑](#footnote-ref-3)
4. It is mandatory to provide an executive summary for each deliverable. [↑](#footnote-ref-4)
5. Indicate Main Author(s) with an “X” in this column. [↑](#footnote-ref-5)
6. Deliverable leader according to FPP, role definition in PCA. [↑](#footnote-ref-6)
7. Person(s) from contributing partners for the deliverable, expected contributing partners stated in FPP. [↑](#footnote-ref-7)
8. Typically person(s) with appropriate expertise to assess deliverable structure and quality. [↑](#footnote-ref-8)
9. Status = “Draft”, “In Review”, “Released”. [↑](#footnote-ref-9)