Information and Communication Technologies (ICT) Programme

Project N°: H2020-ICT-2016-1-732105

D3.6: Cross-layer Modelling Methodology for CPS

Lead Beneficiary:	TNO
Workpackage:	WP3
Date:	23.04.2018
Distribution - Confidentiality:	Public

Abstract:

This document establishes the Modelling Methodology for the CERBERO project. Research on system modelling in CERBERO outlines the main challenges on modelling Cyber-Physical Systems, which arise from the intrinsic heterogeneity, concurrency, and runtime adaptivity of such systems.

Disclaimer

The CERBERO Consortium is the following:

Num.	Beneficiary name	Acronym	Country
1 (Coord.)	IBM Israel – Science and Technology LTD	IBM	IL
2	Università degli Studi di Sassari	UniSS	IT
3	Thales Alenia Space Espana, SA	TASE	ES
4	Università degli Studi di Cagliari	UniCA	IT
5	Institut National des Sciences Appliquees de Rennes	INSA	FR
6	Universidad Politecnica de Madrid	UPM	ES
7	Università della Svizzera italiana	USI	CH
8	Abinsula SRL	AI	IT
9	Ambiesense LTD	AS	UK
10	Nederlandse Organisatie Voor Toegepast Natuurwetenschappelijk Ondeerzoek TNO	TNO	NL
11	Science and Technology	S&T	NL
12	Centro Ricerche FIAT	CRF	IT

For the CERBERO Consortium, please see the http://cerbero-h2020.eu web-site.

Document Authors

The following list of authors reflects the major contribution to the writing of the document.

Name(s)	Organization Acronym
Dr. Rer. Nat. Julio A. de Oliveira Filho	TNO
Dr. Joost Adriaanse	TNO
Dr. Maxime Pelcat	INSA
Dr. Francesco Regazonni	USI
Dr. Francesca Palumbo	UNISS
Dr. Michael Masin	IBM
Dr. Evgeny Shindin	IBM
Dr. Katiuscia Zedda	Abinsula

Document Revision History

Date	Ver.	Contributor (Beneficiary)	Summary of main changes
02.01.2018	0.1	Julio Oliveira	Suggestion for ToC Dump of SoA material
22.01.2018	0.2	Julio Oliveira	Dump of material – no detailed organization yet
20.02.2018	0.3	Maxime Pelcat Francesca Palumbo Francesco Regazonni Michael Masin Evgeny Shindin Joost Adriaanse	Contributions from partners: Models of architecture Hybrid modelling Optimization Cross-layer modelling KPI modelling Agent based modelling For each of these topics, state of the art and contribution of the CERBERO framework.
12.03.2018	0.4	Julio Oliveira	Integration of contributions Text alignment for readability

19.03.2018	0.5	Julio Oliveira	Streamlines the text, brings in introductory paragraphs and executive summary. Integration of last contributions
29/03/2018	0.6	Francesca Palumbo Katiuscia Zedda	Internal Review
11/04/2018	0.7	Julio Oliveira	Integration of review comments Inclusion of section on how to read the document Included table of related requirements
24/04/2018	1.0	Julio Oliveira	Final version – beta. Ready for last internal review.
07/06/2018	1.1	Julio Oliveira	Final version – alpha --. Incorporated all final review remarks and proposed close document writing.

Table of contents

1 Executive Summary

This document establishes the Modelling Methodology for the CERBERO project [CERBERO17]. Research on system modelling in CERBERO outlines the main challenges on modelling Cyber-Physical Systems (CPSs) which arise from the intrinsic heterogeneity, concurrency, and runtime adaptivity of such systems. In this document, we discuss state-of-the-art techniques that address the modelling challenge. Specific technologies discussed in here include modelling for different abstraction levels (from system level to hardware-software implementation), multi-view modelling techniques, model based design space exploration, and modelling of distributed and concurrent systems.

The CERBERO project introduces innovations in each one of the topics mentioned above. In particular, CERBERO's approach for modelling CPSs builds upon:

an integral modelling for different abstraction levels (cross-layer modelling).

an integral multi-view modelling, simulation and analysis, focused on facilitating the exchange of information between partial aspect models and on facilitating the interoperability of design tools.

an efficient model based design space exploration. Including hybrid modelling of computational and physical systems for improving design space exploration capabilities.

an intermediate format for exchanging model information between tools, targeting increase in tool interoperability.

a catalog and characterization of models of computation, in order to guide the generation of new aspect models. Also, on understanding the relationships between different models of computation to guide model transformation and interoperability between models with different computational semantics.

Accordingly, this document discusses these innovations and makes an initial assessment of their impact in the design of CPSs.

1.1 Structure of Document

The suggested way to read this document can be seen in Figure 1. Section 2 opens the technical discussion in this document grounding *modelling* as a design activity and part of an engineering process known as *model-based design*. Within section 2.1, we delineate the view and focus of the research on modelling techniques within the CERBERO project – we point in a summary way all the main contributions that are targeted by the project.

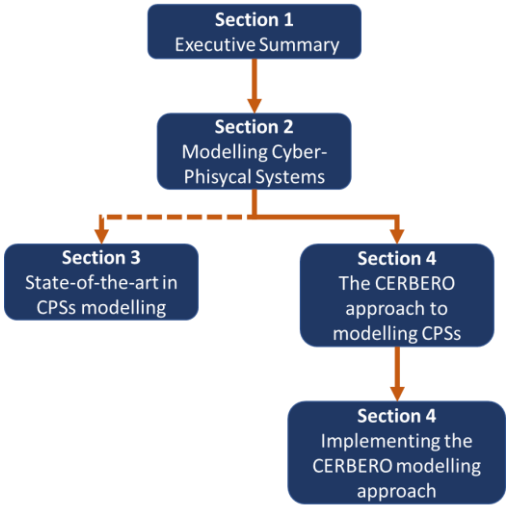


Figure 1: Structure of the D3.6 document

Section 3 reviews the state of the art in several focal points of the modelling activity – those corresponding mainly to the research topics within CERBERO. Aim is to make a

clear statement on what was available in the literature before CERBERO and what contributions of the CERBERO project advances this state-of-the-art scene. **This section can be skipped by a reader that is already up-to-date on the most recent advances in modelling of CPSs**.

Section 4 discusses each one of the research topics and modelling techniques that are further developed in the CERBERO project, with the intention of explaining their fundamental differences to the state-of-the-art and establishing the value added by their innovations.

Section 5 connects all proposed modelling technique to the CERBERO tools and use cases where the respective research and validation efforts take place.

1.2 Related Documents

CERBERO **D3.4 – Modelling of Key Performance Indicators** [CERBERO_D3.4]

The KPIs can be used to represent the system properties. CERBERO proposes innovative techniques to model Key Performance Indicators and it uses them to guide other modelling aspects as well (for example, the choice on Models of Computation). The modelling of KPI is of such importance within the set of modelling innovations introduced by CERBERO, that it is detailed in an apart document (D3.4). For this reason in this document (D3.6), we only highlight the innovations introduced and refer mainly of the technical discussion to the text in D3.4.

CERBERO **D3.5 – Models of Computation** [CERBERO_D3.5]

Similar to the modelling of KPIs, CERBERO proposes many innovations on cataloging and operating models of computations. Due to the details and importance of these contributions, Models of Computation is more extensively discussed in an apart document. For this reason in this document (D3.6), we only h highlight the innovations introduced and refer mainly of the technical discussion to the text in D3.4.

1.3 Related CERBERO requirements

Deliverable D2.7 of the CERBERO project [CERBERO_D2.7] defines a list of CERBERO Technical Requirements (CTRs) the project should achieve. Each of them is referenced with a unique identifier ranging from 0001 to 0020. Research topics related to modelling activities are covered as summarized in Table 1.

Table 1: Links to CERBERO technical requirements

CTR id	CTR Description	Link with the D3.6 document on *Modelling Methodology for CPSs*
0001	CERBERO framework SHOULD increase the level of abstraction at least by one for HW/SW co-design and for System Level Design.	Integral cross-layer modelling Integral multi-view modelling Key Performance Indicators
0002	CERBERO framework SHOULD provide interoperability between cross-layer tools and semantics at the same level of abstraction.	Integral multi-view modelling CERBERO Intermediate Format Key Performance Indicators
0004	CERBERO framework SHOULD provide software and system in-the-loop simulation capabilities for HW/SW co-design and System Level Design.	Integral multi-view modelling Model-based design space exploration
0005	CERBERO framework SHOULD provide multi-viewpoint multi-objective correct-by-construction high-level architecture	Integral multi-view modelling
0007	CERBERO framework SHALL define methodology and SHOULD provide library of reusable functional and non-functional KPIs.	Key Performance Indicators
0009	CERBERO SHALL develop integration methodology and framework.	This document
0020	CERBERO framework SHALL provide methodology and tools for development of adaptive applications.	Key Performance Indicators Model-based design space exploration Models of Architecture

2 Modelling Cyber-Physical systems

Nowadays, electronic monitoring and automation systems are becoming pervasive in almost every aspect of human life. Many traditionally human-controlled activities are now performed by (semi-)autonomous systems that actively sense, decide, and act in place of the humans. Examples range from self-driving automobiles to swarms of robots and factory production lines. Such emerging solutions do not work isolated, but they operate within large scale, complex, dynamical systems. They realize sophisticated signal processing algorithms in distributed configurations, often with feedback loops where physical processes affect computations and vice versa. **This tight integration of computation and physical processes makes these systems unique, and we call them Cyber-Physical Systems or simply CPSs**.

The design of CPSs presents particular challenges. Time becomes a matter of correctness instead of performance, because the time it takes to perform a task may be critical to the correct functioning of the system. Consider for example a self-driving car that has to decide about stopping upon or deviating from an approaching obstacle. In CPSs, many things happen at once, as a complex combination of physical and computational processes occur in parallel. Measuring and controlling the dynamics of these processes by orchestrating actions that influence the processes are the main tasks of embedded systems. Consequently, concurrency is intrinsic in CPS. Many CPS systems are also large in the number of participating components and often these components are spread apart, interconnected with very diverse network topologies. During design, the structural aspect of CPSs become as important as the functional aspects.

USE OF MODELS IN CPS DESIGN - ADVANTAGES

The absolute most accepted approach for the design of CPS is by using models – a strategy called **model-based design**. Working with models has major advantages.

Models can be made formal and mathematically/logically sound. We can say definitive things by using models. For example, we can assert that a model is deterministic, meaning that given the same inputs it will always produce the same outputs. No such absolute assertion is possible with any physical realization of a system. If our model is a good abstraction of the physical system (here, "good abstraction" means that it omits only unnecessary details), then the definitive assertion about the model gives us confidence in the physical realization. Such confidence is hugely valuable, particularly for embedded systems where malfunctions can threaten human lives. Studying models of systems gives us insight into how those systems will behave in the physical world.

Additionally, using models in modern engineering became very attractive from an efficiency and economic point of view. **Models are faster and cheaper to construct and easier to manipulate than the real (physical, full-scale) artifacts** they describe. Computer models are used in many engineering disciplines to analyze and predict the behavior of the systems they describe. Engineers use dedicated software tools to interactively create and manipulate the models and to simulate/evaluate the behavior of the systems using various test scenarios. In experimental setups, models can be subjected to

stimuli and conditions that would not be feasible or just be too dangerous to carry out with the real artifact.

Finally, working with **models strongly enables design automation**. (Formal) Models can be checked for language conformance, internal consistency and completeness, and ultimately can be used to analyze the properties of the systems they describe. When these models are made machine readable, software tools can be constructed to manipulate the models and to perform automated operations on them like checking the consistency, constraints and completeness of the model and execute internal model transformations. This offers a great benefit, since carrying out these operations by humans would be too error-prone or could simply be not feasible due to the size or complexity of the model.

MODELLING AS A DESIGN ACTIVITY

We call "**modelling**" **to the set of activities related to writing, manipulating, and transforming models.** The task of the designer during the modelling is to express unambiguously particular properties and behavior of a (sub) system or process we are interested in while neglect others, which are considered irrelevant for a purpose. Some aspects are intentionally omitted to keep the models from becoming overly complex and because the associated component interactions do not play a dominant role in system behavior. There are various model types such as physical, functional, analytical, causal, etc.

Modelling is usually carried out by using a modelling language that consists of a set of modelling primitives with well-defined semantics and composition rules. Using modelling languages, designers are able to create a machine readable specification of the system whose consistency and completeness can then be checked automatically. Modelling languages can be given a textual as well as a graphical representation. Mostly, designers prefer to work with graphical presentations of a model, such as a set of diagrams. Graphical metaphors are used that closely match the abstractions used by the designer when conceptualizing the system. Also, various relations between system components can be shown explicitly in the diagrams. A mixed form of model presentation is also possible, in which parts of the model are expressed using graphical elements and other parts are specified using a textual formalism.

Furthermore, **models can be used both for analysis purposes as well as for synthesis purposes**. If the model semantics allow it, important system properties could be derived early in the development cycle and the designer can reason about and experiment with alternative designs at the abstraction level of the model. This is an enormous advantage over traditional development approaches, where the system is often coded first and where the critical system requirements are met afterwards by tuning and tweaking the system's implementation. In some cases, if the semantics of the model is sufficiently complete, the implementation of the system (software) can be synthesized from the model.

Modelling is an essential activity in the engineering process. **Modelling is the way to operationalize – conceptually and mathematically – certain design steps: design conception, design evaluation and design adjustment steps**, the so called "build – evaluate – adjust" cycle (Figure 2).

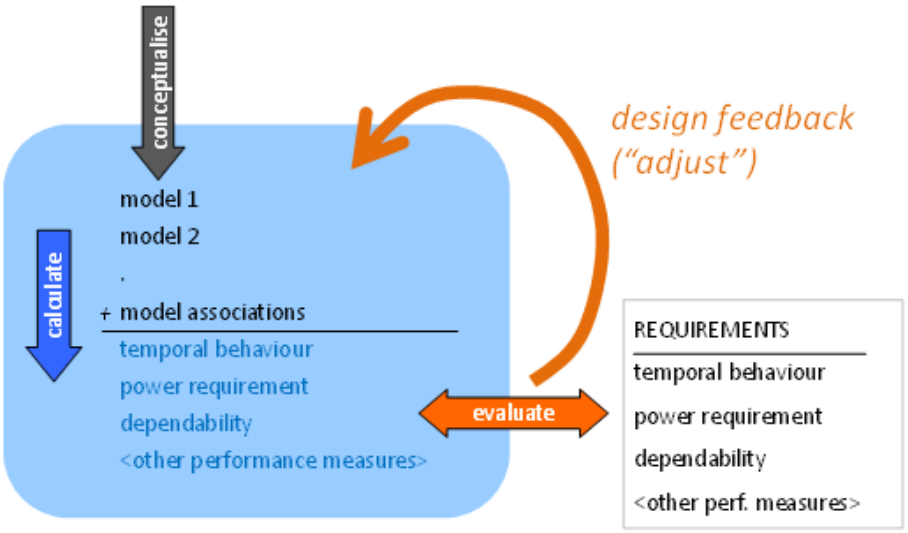


Figure 2: Model-based design cycle

In the **design conceptualization phase, various models are built describing different aspect of the design and their interactions**. From these models, system level (or emerging) properties are derived, that can directly be compared to the (non-functional) requirements. If the result of the evaluation is not satisfactory (structural or parametric) changes in the design are necessary – which in the model-based design approach means manipulating the models.

The reduction of the number of design iterations (inside and across the design stages) can be facilitated by well-informed design decisions, i.e. by reducing the number of incorrect decisions resulting in "backtracks" in the design process. The model-based approach directly supports achieving this goal. The models can represent all relevant knowledge we possessed at a particular point in the design process. With suitable **model evaluation tools information can be derived, that can directly guide the design decision**. The same applies when runtime adaptivity is considered: models lend the system the formal foundation for applying automated reasoning processes to derive adaptation plans.

As the design progresses, more and more details are added to the models, and hence, a more accurate evaluation of the design becomes possible. The **gradual model refinement supports iterative design processes, where the iterations work on increasing level of detail/accuracy**. First only a "rough" design is made (e.g. just identifying the main system components and their connection topology), later the components are detailed and design on a finer granularity is produced. This process can be continued until implementation details (e.g. program code) are added, i.e. blurring the border between system design and implementation.

In the next section, we provide an overview on how the CERBERO project puts together a model-based methodology for the design of CPSs.

2.1 The CERBERO Modelling Approach for Cyber-Physical Systems

The CERBERO project strongly builds upon a model-based engineering approach. CERBERO's modelling methodology builds upon established and validated design practice used in the design of large, networked, embedded systems. **CERBERO** does not

introduce a completely new engineering process, but instead leverage on the best methodologies found in the design community and **extends and improves many parts of the existing modelling and design methodologies.** In fact, almost every activity and research topic within the project is related or based on modelling or manipulation of models. The CERBERO community believes that models may provide an unambiguous, formal, and mathematical base for designing and realizing complex systems.

Figure 3 depicts how the CERBERO project conceives the design of a system. As for engineering process, **CERBERO aims to transform the traditional V-Model approach (MBE) by providing a continuous, short-cycled, incremental design environment (CMBE) enabling early-stage analysis, optimization, fast deployment, and verification of functional and non-functional requirements**.

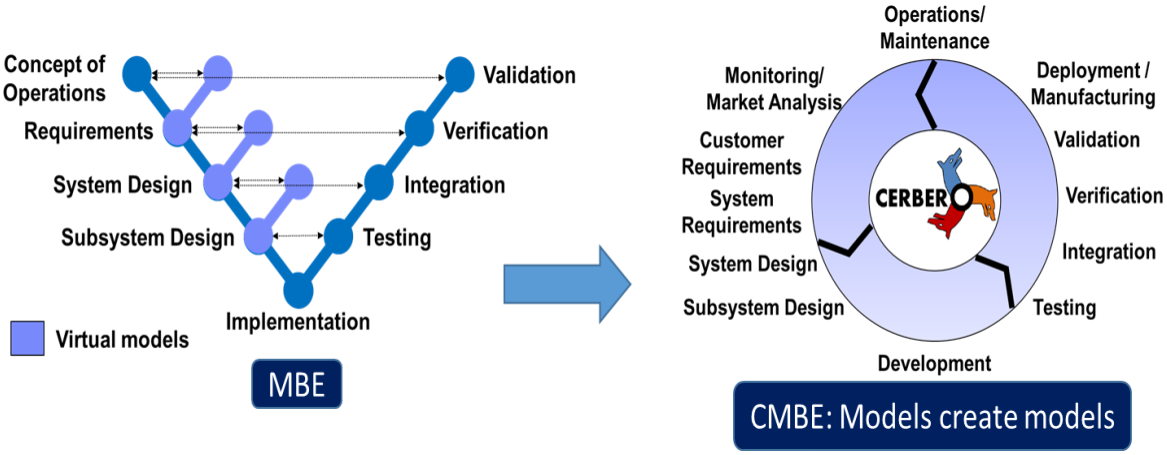


Figure 3: The V-Model engineering process when using a model-based design approach

First, models are used as early as possible in the business development process to formalize user requirements into a system specification. In CERBERO, **models at the most initial design phase compose the specification of the system**. **The following design steps consists in gradually refining, and enriching these models towards an (automatic) physical implementation.** So, for example, the specification models are enriched for an early system-level simulation, analysis, and design space exploration – facilitating an guiding the design decision process. As a result, the requirement level of the models is refined to a functional and logic breakdown of the system using rich domain-specific details. HW/SW co-design and synthesis from high-level of abstractions (semi-)automates the transformation of these models into a physical realization. At all moments, models are also the base for validation and verification at the same level of details. Each iteration of the design process tries to deliver a version of the system: at initial cycles, such deployments are made **onto executable models;** as more design cycles are added, deployments include an interoperable setup with models and real system parts (for example, working with a system-in-the-loop simulator); finally as the design phase approaches its end, deployments tend to assume form of source code or hardware/physical components.

The vision above extends and enriches many other design research groups [PTOL][SEI][VINC12] who build upon similar principles and goals. **CERBERO is**

unique in the several different innovations – many of them related to the modelling activity – proposed to this engineering process. CERBERO innovations target three main areas of contributions:

1) modelling and design of complex, large scale, networked systems;
2) modelling concurrency and distributed behavior;
3) efficient methods and tools for model-based design space exploration, using sophisticated techniques such as modelling of hybrid systems and uncertain environments.

In the next, we highlight each one of the activities that leads us to a unique approach and indicates the detailed discussion of each topic within this document.

MODELLING COMPLEX SYSTEMS

CERBERO modelling methodology brings four innovative contributions to the design of complex systems, as depicted in Figure 4:

(1) **CERBERO innovates with a cross-layer modelling approach where model information flows from more abstract, system-level models into implementation/domain specific models at HW/SW level**. The state of the art in this topic is presented in Section 3.1.1 and our approach is discussed in Section 4.1.1.

(2) **CERBERO build simulation tools based on multi-view (multi-view) modelling approach**. The idea is to conceive a complex system model as a cooperation of many modelling viewpoints, each of which abstracts the system for a purpose, but complements the information in the total. The state of the art in this topic is presented in Section 3.1.2 and our approach is discussed in Section 4.1.2.

(3) **CERBERO proposes a new intermediate format to exchange modelling information between tools**. The CERBERO intermediate format is able to solve many interoperability problems existent in the actual metamodelling oriented approaches. The state of the art in this topic is presented in Section 3.1.3 and our approach is discussed in Section 4.1.3.

(4) **CERBERO proposes a way to model key performance indicators in such a way that KPIs can be more easily re-used among projects and tools, and that enables easier automation of modelling analysis tasks**. This topic is extensively discussed in deliverable D3.4 [CERBERO_D3.4], but as it is related to the modelling activity, we briefly discuss it in here and highlight our current achievements. The state of the art in this topic is presented in Section 3.1.4 and our approach is discussed in Section 4.1.4.

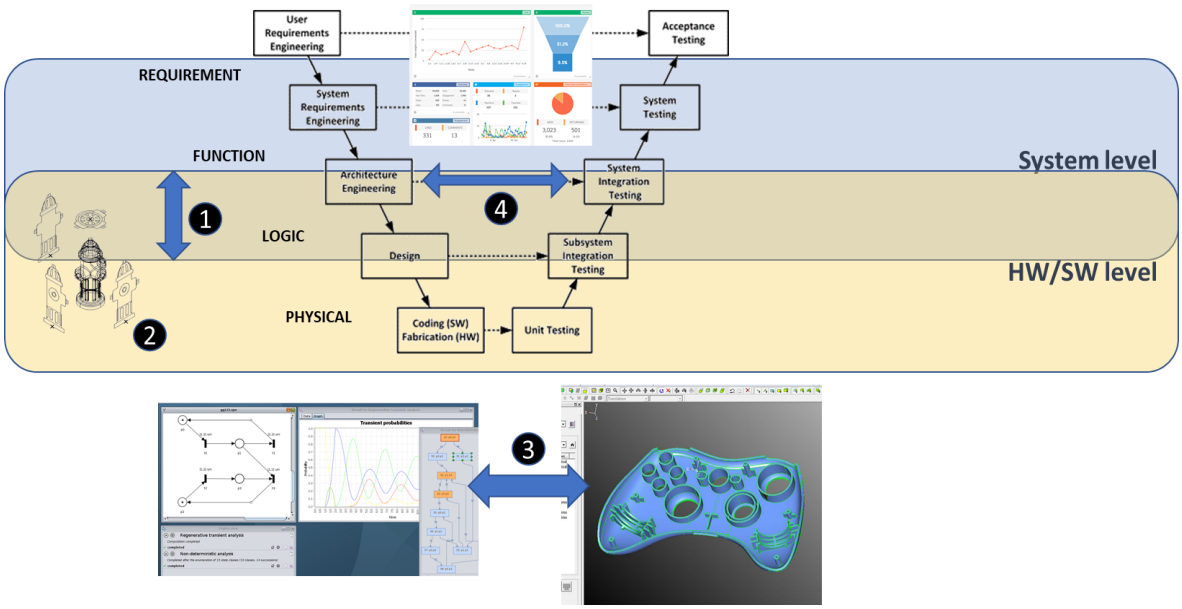


Figure 4: Innovations of CERBERO in the modelling of complex systems. (1) Cross-layer modelling, (2) Multi-view based simulations, (3) easier interoperability between tools due to the CERBERO Intermediate Format, and (4) modelling of key performance indicators.

Modelling Concurrency and Distributed Behavior

Concurrency is an intrinsic characteristic found in most of systems targeted by CERBERO. Concurrency refers to the ability of different parts of a system to execute out-of-order (or concurrently) without affecting the outcome. Such characteristic is dominant in modern systems due to their non-locality, large size, and non-centralized organization.

Modelling concurrent systems has always been a challenge, but much progress has been made in recent times due to the formalization of communication and behavioral models by means of models of computation. **CERBERO invests in elaborating a catalog of models of computations and a methodology for choosing an appropriate Model of Computation (MoC) for the purpose in hand**. As an example, suppose a designer is deciding to model the states of the system and its transitions. A finite state machine (MoC) is appropriate in this case only if the system has no fork/join mechanisms (or making them explicit is not relevant for the aspect to be modeled). Otherwise, using Petri-Net formalism (MoC) would be more adequate on exposing these concurrency mechanisms.

This topic is extensively discussed in deliverable D3.5 [CERBERO_D3.5], but as it is related to the modelling activity, we briefly discuss it in here and highlight our current achievements. A short overview of the state of the art in models of computation is given in Section **Error! Reference source not found.** and the CERBERO innovations at this area a re shown in Section **Error! Reference source not found.**.

Models and Tools for Design Space Exploration

One of the strongest benefits of using models is the possibility to explore different design options at a lower cost without building different prototypes. As discussed before, models are easier to modify, and their mathematical background allows to reason on the outcome of their implementations. But design space exploration may be a difficult challenge due to

the design space explosion problem – there are too many variants of a model (even small ones) to be evaluated efficiently.

CERBERO introduces powerful techniques for the design space exploration of complex system models, considering adaptive behavior, hybrid systems, and uncertainty intrinsic to the environment where the system must work on. In CERBERO, we strive to describe large combinatorial problems as hybrid models – mixing discrete and continuous modelling. Such representation of a design space exploration problem can speed up the search in the design space. An overview of design space exploration techniques is given in Section 3.3**.** Innovations introduced by CERBERO are discussed in Section 4.3.

USE AND VALIDATION OF THE CERBERO MODELLING METHODOLOGY

Within the CERBERO project, all the innovations proposed for the design and modelling phase of a system are assimilated into tools that make part of the CERBERO framework. The introduced techniques are then validated by applying the tools in the design of the CERBERO use cases. In Section 5, we discuss a mapping between each proposed innovation, the tools where they are incorporated, and the use case within CERBERO where they are used and validated.

3 Survey on Modelling Cyber-Physical Systems – challenges and current approaches

This section will review the state of the art in modelling and model-based design in CPSs. But, as discussing all aspects of modelling could become extremely extensive – going from modelling languages all the way down to code generation – we will focus on the state of the art in the topics related to the research inside CERBERO. In this section we frame the actual development status in the CPSs community, and we use the same topic structure in Section 4 as a background to show where CERBERO innovates.

3.1 Modelling complex systems

Dealing with complexity is intrinsic in the modelling of CPSs. Such complexity does not come only from the size of the systems – sometimes CPSs designs can be quite small – but specially from the multi-disciplinary nature of such systems. CERBERO invests in some focal points to cope with a complex design:

(1) enable model information to propagate from high levels of abstraction towards lower levels and implementation in a more natural way – we call that cross-layer modelling (or design);

(2) enable the model information to propagate between models (viewpoints) in a more natural way – we see that as an exercise on multi-view model-based design

(3) the exchange of model information in topics (1) and (2) must be operationalized at tool level to leverage in automation and correctness of model transformation methods

(4) finally, we invest in identifying universal ways to model key performance indicators, such that results can be more easily compared and reported.

In the following, we discuss the current state of the art in the literature and about these topics.

3.1.1 Cross-layer modelling

The intrinsic complexity of CPSs is the recipe of their large potentials: interconnecting what in the past have been separate systems certainly open a plethora of new possibilities but, at the same time, it comes at the price of increased the design and verification challenges. At the root of this issue there is our inability to rigorously model the interactions between the physical and the cyber sides.

The systems modelling language (SysML) [OMG12] provides a general-purpose notation for systems engineering, being capable of supporting systems that present hybrid phenomena, where continuous (suitable for physic components) and discrete (suitable for cyber components) time models mix. This aspect makes SysML suitable to be used in CPS design environment. SysML can be defined as semi-formal language: it has a formal syntax, but no formal semantics. The idea is being able to support different types of systems, by choosing the appropriate semantic. Semantics represent also the instrument to support co-simulation.

The INTO-CPS project [INTOCPS] aims at creating an integrated "tool chain" for comprehensive Model-Based Design (MBD) of CPS. Among the other features, INTO-CPS provides support for the holistic modelling of CPS based on a SysML profile (proposed in the project) with a formal semantics for CPS. Such a profile leverages on a subset of SysML notations (block definitions and internal block diagrams) and is meant to target multi- and heterogeneous modelling and co-simulation. INTO-CPS supports co-simulation leveraging on Functional Mock-up Interface (FMI) standard [FMI14]. FMI wraps models from different tools and abstractions in Functional Mock-up units (FMUs), enabling inter-FMU communication and importing into hosting tools. Models are black boxes, FMU compliances ensures protection of the modelled IPs, as well as, interoperability.

Other approaches, to provide model simulation, generate code from SysML. Café et al [CAFE13] address the cross-layer modelling problem by generating co-simulation models (SystemC-AMS, which provides pre-built MoCs allowing co-simulation of continuous and discrete components) from different MoCs. Wawrzik et al [WAW15] translate SysML into SystemC to enable the simulation of the modelled CPS, using specific dialects of System-C designed to tackle the phenomena being modelled (e.g. hardware/software, network and propagation, and analog and physical processes).

Another interesting approach to handle the modelling complexity of CPS designs is provided by the "contract-based" approach [VINC12], which is intended to be used at all stages of system design (from requirements capture to embedded computing development) to properly deal with the integration and composition of heterogeneous components and hybrid environments. Contracts explicitly handle pairs of properties, respectively representing the assumptions on the environment and the promises of the system under these assumptions. They can be handled to address non-functional properties during the system design and to structure communication in a multidisciplinary field as the CPS one, where teams having diverse backgrounds and different fields of expertise must cooperate. In this case contracts allow defining clear interfaces between the different disciplines. In [CANC15], Cancila et al. specified a domain specific language tailored to contract-based design as a SysML profile to apply contracts over block diagrams.

Ptolemy II [PTOL] provides actor-oriented modelling which supports multiple domains, being able to model them in the same design workspace. One of the basic assumptions behind Ptolemy is that modelling the diverse implementation technologies and their interaction is not reasonable within a homogeneous environment. Heterogeneous modelling is used to provide cross-layer support. In the Ptolemy environment actors could be set to various MoCs to represent either physical and/or cyber components. The interaction among MoCs leverages on the object-oriented principles of polymorphism and information hiding. For example, using Ptolemy software, a high-level dataflow model of a signal processing system can be connected to a hardware simulator that in turn may be connected to a discrete-event model of a communication network.

In CERBERO we are developing an integral approach for modelling different abstraction levels and different views of the CPS. As it will become clearer in the following sections, we intend leverage on an intermediate format to allow sharing information among levels and views. One fit to all solution would certainly do not answer to the peculiarities of the different layers of the system. In the CERBERO project, different Model of Computations and Models of Architecture (see Sections 4.2.1) are exploited to represent all the different views, layers, and components of a CPS. The CERBERO Intermediate Format (CIF) purpose is to make them talk appropriately leading us toward a cross-layer approach. The goal is to make the CIF suitable to represent the CPS and the KPIs that describes it and, at the same time, to make each tool in the CERBERO framework stack (from the verification layer down to hardware) capable of reading and manipulating the CIF, and to send feedbacks to other tools through it when needed.

3.1.2 Multi-view model based design

When using model-based design methodologies formal modelling languages can be defined such that they allow the description of a target system using multiple system views. A system view, or system aspect, is a way to look at or describe a system as a whole: **each system view has its own associated semantic domain and can provide an exhaustive description of the system, but only from that point of view**. Different groups of users of a system may consider completely different aspects of that system. For example, an accounts clerk will have a completely different view of the companies' administrative system than its system developer. **Using multi-view modelling framework provides several advantages**:

- It allows to capture the different ways separate groups of users of a system view that system. Each group of users or stakeholders, has its own concerns with respect to the system to be realized, possibly expressed as a set of requirements and/or key performance indicators in a semantic domain (e.g. responsiveness, throughput, energy consumption, dependability, etc.).
- The development of a system typically involves the cooperation of multiple design disciplines. Each discipline will typically be addressed by only a subset of experts of the team. Design decisions made in one discipline can have consequences in other disciplines. Using multi-view models and the relevant analysis tools make this kind of interdisciplinary design trade-off manageable.
- A multi-view modelling language will improve the usability of the models and offer greater flexibility in the exploration of design alternatives as the different system aspects in the model can be manipulated more independently. The models subsequently can be used for many purposes that can aid a system architect, such as automatic code generation, design optimizing, system evolution etc. [KARSAI10]. The interaction between the different models then plays a crucial role in the design process.

- It is a very powerful means to reduce model complexity: it enables designers to focus on only one system aspect at a time. By starting at the more abstract system aspects, the system design effort can progress in stages, where at each subsequent stage more detail of the system components is added, and more component interrelations are captured by the model.

Many approaches to multi-view modelling can be found it the literature. Some of them target specific application domains, while others are more general purpose. For example, RM-ODP (Reference Model-Open Distributed Processing), a reference model introduced in the eighties as the result of a cooperative effort by the ISO (International Standards Organization) and ITU-T (International Telecommunication Union) [ISOIEC]. RM-ODP provides a framework through which analyzing, describing and specifying a system from different perspectives, called viewpoints.

Another example is the Architecture Analysis and Design Language (AADL), which was standardized by the Society of Automotive Engineers (SAE) [FEILER12]. AADL defines a language for describing both the software architecture and the execution platform architectures of performance-critical, embedded, real-time systems. An AADL model describes a system as a hierarchy of components with their interfaces and their interconnections. AADL components fall into two major categories: those that represent the physical hardware and those representing the application software.

SysML is a general-purpose modelling language for systems engineering that supports the specification, analysis, design, verification and validation of a broad range of complex systems, including hardware, software, information, processes, personnel and facilities [SYSML]. It uses a subset of UML 2.1 and provides additional extensions needed to fulfill the requirements for the modelling language specified by the SE DSIG (Systems Engineering Domain Special Interest Group) of the OMG.

In CERBERO we are researching how to write model viewpoints in such a way that they become complementary: easy to understand and manipulate when used apart, but expressive and rich when combined. The way CERBERO approaches such challenges is to develop extra viewpoints (such as a mapping viewpoint) which is in itself a model on how two other viewpoints should be connected and combined. This way of modelling is largely used in the CERBERO tool framework DynAA, and will be explained in more details in Section 4.1.2.

Cross-layer and complementarities of viewpoints are the instruments we foresee to go beyond the traditional separation of concerns. Most properties and behavior of the whole CPS are emergent, i.e. they cannot be simplistically inferred from those of the individual components. Classical separation of concerns, despite being extremely useful, may lead to miss important interactions, which is why it has to be enhanced to go beyond boundaries.

3.1.3 Interoperability between model-based design tools

The model-based design of large complex CPSs heavily depends on tooling. Tooling is necessary to create the models and operationalize its manipulation: model checking, simulation, code generation, modifications, analysis, etc. Moreover, proper model-based cross-layer and multi-view design – such as in CERBERO – depends on the good **interoperability between** these **tools: tools should work as much as possible by enriching, modifying, and transforming the same system model.** Constantly (and manually) re-writing the same models every time a new tool is needed leads to errors and misinterpretations.

Providing a common base – or intermediate format – for sharing models between tools is the key to interoperability, but also a difficult challenge. Interoperability means that the information from one model should be accessible, transferable, and possible modifiable to/from other models (and their view points). This section, gives a short overview on the state of the art on representing and exchanging information between tools.

In literature, the most established approach for representing model information is the use of metamodels. **A metamodel defines formally the concepts allowed to be present in the model and the rules and relationships between these concepts**. In other words, metamodels provide a formal organization to the information in a model. In the literature, the most well-known frameworks to describe metamodels (and automatically generate modelling tools) are the Universal Modelling Language (UML)[UML] and its many flavors (e.g. SYSML]), the Eclipse Modelling Framework [EMF], MetaCase's GOPRR [VLAD12] [KERN11] [METAEDIT], and [GME]. UML can also be used as metamodelling languages, but this is not a very common practice.

As an example, we depict in Figure 5 a simplified version for the metamodel of the task view in the CERBERO framework tool DynAA (see section 4.1.2 for more details on the task or functional view). The task view represents the functionality and process level concurrency in the system. Such a metamodel allows a designer to produce a whole family of models, each of which describing the tasks in a system and their input/output ports, and the links representing communication between the tasks. If a model conforms to the metamodel, such as the example above, it can be represented by a graph. The nodes of this graph are the concepts allowed in the model by the metamodel, whereas the edges of the graph describe the relationships between these concepts.

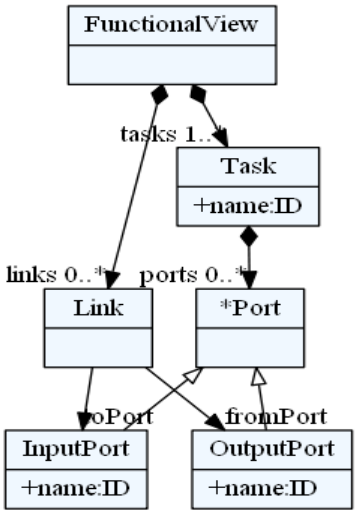


Figure 5: Metamodel for the task view in the CERBERO framework tool DynAA

The metamodel approach is very strong and valuable. When a metamodel is known, we can build tools to collect, organize, store, and manipulate the information of a model. By traversing the model graph, a tool can 'understand' the model, access its concepts and check which are the specific relationships in the model. The metamodel approach became specially strong and important because it enabled the automated generation of design tools. Given a metamodel, it is possible to automatically

generate modelling tools, formal verification tools, code generators, analysis tools, and simulators for all the family of models that conform to the metamodel.

Despite all the advantages of the **metamodelling approach**, it **faces serious challenges in three situations** related to the interoperability of tools [PARSONS]:

1. **The multi-view interoperability problem**: The complete system model has to be defined by many views (see explanations in sections 3.1.2 and 4.1.2). Creating a modelling framework flexible enough to accommodate all views and potentially new (not yet considered ones) is what we call *the multi-view interoperability problem*.
2. **The multi-tool interoperability problem**: The system model (or part of it) must be shared by many generic tools. They reflect the multi-view problem in a tool operational environment.
3. **The model maintenance problem**: The system model must be consistent and maintained through evolving versions of the metamodel, evolving versions of the intermediate/persistence format, and multi-versions of the tools.

All three situations are natural on the design of CPSs and therefore on the field of innovation within CERBERO. Proposed approaches to solve these problems are not successful and only partially alleviate the problems. Table 2 shows, for each interoperability problem, the unaddressed challenges:

Table 2: Unsolved interoperability problems when using metamodels

	Multi-view interoperability problem	Multi-tool interoperability problem	Model maintenance problem
Unified metamodels	Model information used by different views are either duplicated – generating several data management problems – or kept in one of the views, what forces other views to incorporate knowledge about other metamodels.	Tools are forced to comply to a large, not flexible enough standard. It creates mostly adoption barriers. Also, tools trying to read/understand an intermediate format are often pushed to be compliant to all the metamodels instead of only the part that is interesting for their modelling purposes	Evolving the metamodel of one view is likely to affect other metamodels.
Independent metamodels per view	Relies strongly on tool automation to make it easy dealing with so many different metamodels.	Tool developers are resilient to implement details of third-party metamodels in their tools if they do not see the market payoff	No unified method to describe versioning in different metamodels.

The CERBERO project proposes (and investigates) new ways to define the intermediate format between tools to solve the interoperability problems mentioned above. In summary, the CERBERO project propose a two-layered intermediate format that decouples the model information (content) from the way the information is represented (schema). This approach is innovative, follows modern developments in non-schema databases, and promises to bring advantages in three aspects:

1. *model information can be easier shared by multiple views;*
2. *tools do not have to deal with metamodels of other tools; and*
3. *information representation is ready for dealing with versioning systems.*

We detail the CERBERO intermediate format proposal, and its relation to the modelling activity in section 4.2.2.

3.1.4 Modelling Key Performance Indicators

Adopting a cross-layer and multi-view modelling framework (see Sections 3.1.1 and 3.1.2) provides rich capabilities to analyzing, communicating, and documenting design choices; but this is merely the first step. When realizing a system, designers often have a very large space of possible alternatives. The selection of the most suitable alternative is usually a multi-objective problem, which aims at identifying the system capable of globally maximizing the design goals. It becomes necessary to efficiently explore alternatives and evaluate the design alternatives.

Such exploration can also occur in real time, in which the system searches during runtime for a more adapted or appropriate configuration. Such feature ultimately enables adaptivity, a fundamental requirement for current and future generations of CPSs. Ideally, **design alternatives should be characterized in such a way that the derived properties should directly be comparable to key performance indicators (KPIs).**

KPIs are a well-known concept from economics and management [ROUB13] [ADEL09], where are used to evaluate the performance of an organization. A similar concept can be applied to CPSs. KPIs must be selected in a way that they will define the goal of the systems. KPI measures are mainly the output of design evaluation and will allow to quantify the discrepancies between the system goals and the actual or estimated performance. Along the design process and during the whole lifetime of the system the system designers (and the adaptive systems themselves) must make informed decisions when selecting the most “promising” design/configuration alternative. The selection should be driven by quantified properties of the design. These properties are originated in the design of components, compositions, parameters; and in the execution scenarios, i.e. the interactions between the systems designed and its embedding environment.

The model-based engineering approach formalizes all relevant aspects of the design in models and thus gives the formal foundation for deriving the emerging properties of the design. Frequently, the quantified design properties are aggregated in a "design quality measure" and used to guide a constrained design optimization process. The model-based derivation of the design properties is just a manifestation of old and established engineering approach, namely use models to predict system behavior [MORIN09].

The model-based derivation of design properties and its use in "evolving" the system go beyond strictly design-time activities [KARSAI10]. The driving forces behind system evolution are "keeping operational" or "making it better" the system implemented as expressed in a quality measure. In runtime reconfigurable designs the calculation of the emerging system properties is carried out during the nominal operation of the systems to detect anomalies and consequently initiate and guide redesign (optimization) in runtime. Due to the possibly prohibitively large design space and the complexity of the design process the scope of the runtime redesign (i.e. the monitored set of key performance indicators and the investigated design alternatives) should be constrained [STRE06].

The main innovation of CERBERO is to use KPI analysis to manage both the design phase and the intelligent adaptation of complex cyber-physical systems. We start from model based KPI analysis as a way to guide the exploration of design alternatives. Then we extend the models of KPIs to guide real time adaptation of the system. Models of KPI in CERBERO will be cross-layer, namely that each model can be refined at a different level of abstraction, or mapped to a different layer, of the system. This would allow to trade the precision of the KPI evaluation with the required adaptation performance of the system.

3.2 Modelling reconfiguration and self-adaptation

Reconfiguration and self-adaptation is based on modelling of concurrency and intrinsic uncertainty in behavior of CPS and its environment. In the next sections we describe state of the art in their modelling.

3.2.1 Models of Computation

Complexity in CPSs also come due to the intrinsic concurrency characteristic of these systems – typically distributed, networked, dynamic, and adaptive. Though many progress has been made in the field of designing concurrent systems, many other open questions remain, especially on the analyzability and expressiveness of concurrent models. CERBERO's research tries to cope with modelling concurrency based on models of computation and agent-based software. We discuss the state of art in these topics in the following.

A Model of Computation (MoC) [LEE98] defines the semantics of a computational system model, i.e. which components the model can contain, how they can be interconnected, and how they interact. Every programming language has at least one (often several) underlying MoCs. A MoC describes a method to specify, simulate and/or execute algorithms. MoCs were much promoted by the Ptolemy and Ptolemy II projects from the University of California Berkeley. In [CHANG97], Chang, et al. explain how several MoCs can be combined in the Ptolemy tool. MoCs can serve three different purposes:

1. **Specification**: A specification model focuses on clearly expressing the functionalities of a system. It is especially useful in standards documents.
2. **Simulation**: A simulation model is used to extract knowledge of a system when the current implementation is not available. It may be much simpler than the final code and is focused precisely on the features of interest.
3. **Execution**: An execution model must contain all the information for the final code execution.

The definition of MoCs is broad and covers many models that have emerged in the last few decades. The notion of a MoC is close to the notion of a programming paradigm in the computer programming and compilation world [VANROY 09]. Arguably, the most successful MoC families, in terms of adoption in academic and industry worlds are [PELCAT13]:

- Finite State Machine MoCs (FSM) in which states are defined in addition to rules for transitioning between two states.
- Process Network MoCs (PN) in which concurrent and independent modules known as processes communicate ordered tokens (data quanta) through First-In First-Out (FIFO) channels.
- Discrete Event MoCs (DE) in which modules react to events by producing events.
- Functional MoCs in which a program does not have a preset initial state but uses the evaluation result of composed mathematical functions.
- Petri Nets which contain unordered channels named transitions, with multiple writers and readers and local states called places, storing data tokens.
- Synchronous MoCs in which, like in Discrete Events, modules react to events by producing new events but contrary to Discrete Events, time is not explicit and only the simultaneity of events and causality are important.

Within the CERBERO project, several studies are on-going that aim at extending MoCs to provide CPS-friendly features. These studies include for instance the extension of dataflow to represent persistent states that hinder parallelism and limit system scalability, integrating some information on system non-functional properties into a MoC to cross the barrier between requirements and abstract application modelling, enabling "moldable" parameters that can deeply change the nature of an application while still making design-time analysis and runtime management possible. More details on the CERBERO evolutions of MoCs are given in deliverable D3.5.

3.2.2 Modeling of uncertainty of CPS and operational environments

The data of real-world problems more often than not are uncertain - not known exactly at the design time. The reasons for data uncertainty include, among others: measurement/estimation errors coming from the impossibility to measure/estimate exactly the data entries representing characteristics of physical systems/technological processes/environmental conditions, etc.; implementation errors coming from the impossibility to implement a system exactly as it is designed. Moreover, real-world applications cannot ignore the possibility that even a small uncertainty in the data can make the nominal optimal solution to the problem completely meaningless from a practical viewpoint. Fortunately, there are design techniques that are developed to overcome this issue and can be used for CPS reconfiguration and self-adaptation.

Robust Optimization (RO) offers a methodology capable of detecting cases when data uncertainty can heavily affect the quality of the nominal solution, and of generating a robust solution immunized against the effect of data and model uncertainty [Ben-Tal – El Ghaoui – Nemirovski, 2009]. The uncertain numerical data belonging to a given uncertainty set could be separated from the certain problem structure (i.e., goals, constraints, and decision variables). In its original form, RO dictates that constraint must be satisfied for all possible realizations of uncertainties. However, the more uncertainty we should deal with, the more constrained the design will be and the value of the objective function of lesser quality. To provide more efficient solutions, RO was extended to take into account the probability of meeting a constraint in the form of Chance Constraints. When using CC, we ask RO to provide a solution ensuring that the chance of not meeting a constraint is less than ϵ instead of meeting the constraints under all circumstances. As some uncertainties are "realized" and more information become available, one could prefer to find not an optimal control but optimal control policy that leads system reconfiguration based on the realization of the uncertainties. Affinely Adjustable Robust Counterpart (AARC) is class of tractable approaches of RO policies where the future control depends linearly on the realized uncertainties. Note, that when such a policy is realized as a part of system architecture it can be viewed as self-adaptation policy and thus this optimization method extends self-adaptation techniques. For an example of application of robust optimization to system design see [Shindin et al, 2014]

Stochastic programming models assume that uncertain parameters have known probability functions. The goal of stochastic programming is to find some policy that is feasible for all (or almost all) the possible data instances and maximizes the expectation of some function of the decisions and the random variables. More generally, such models are formulated, solved analytically or numerically, and analyzed in order to provide useful information to a decision-maker. The most widely applied and studied stochastic programming models are two-stage (linear) programs. Here the decision maker takes some action in the first stage, after which a random events occur, affecting the outcome of the first-stage decision, and the second stage decisions are made.

Two-stage model was originated in the works of Beale [Beale, 1955], and Dantzig [Dantzig, 1955]. The classical two-stage *linear* stochastic programming problems can be formulated as: $\min_{x \in X} g(x) = c^T x + E[Q(x,\xi)] \quad s.t.\ Ax = b, x \geq 0$ where $Q(x,\xi)$ is the optimal value of the second-stage problem $\min_{x \in X} Q(x,\xi) = q(\xi)^T y \quad s.t.\ T(\xi)x + W(\xi)y = h(\xi), y \geq 0$ The difficulties in solving this problem mainly concern the

computation of which typically represented by the multiple integral, wherein the evaluation of the integrand requires the solution of a linear programming problem (LP). To solve the two-stage stochastic problem numerically, one often need to assume that the random vector ξ has a finite number of possible realizations, called scenarios, say $\xi_1 \dots \xi_K$, with respective probability masses $p_1 \dots p_K$. Then the expectation in the first-stage problem's objective function can be written as the summation. The above numerical approach works reasonably well if the number K of scenarios is not too large. In practice it might be possible to construct scenarios by eliciting expert's opinions on the future. The number of constructed scenarios should be relatively modest so that the obtained deterministic equivalent can be solved with reasonable computational effort. Another method to reduce the scenario set to a manageable size is by using Monte Carlo simulation. Suppose the total number of scenarios is very large or even infinite. Suppose further that we can generate a sample $\xi^1 \dots \xi^N$ of N replications of the random vector ξ. Usually the sample is assumed to be independent identically distributed. Given a sample, the expectation function $Q(x, \xi)$ is approximated by the sample average $\frac{1}{N}\sum_{j=1}^{N} Q(x, \xi^j)$ This formulation is known as the *Sample Average Approximation* method. The SAA problem is a function of the considered sample and in that sense is random. For a given sample $\xi^1 \dots \xi^K$ the SAA problem is of the same form as a two-stage stochastic linear programming problem with the scenarios $\xi_1 \dots \xi_K$ each taken with the same probability $p_k = \frac{1}{K}$. More details and algorithms concerning numerical methods can be found in [Ermoliev-Wets, 1988].

The two-stage stochastic programming models have been static in the sense that a (supposedly optimal) decision can be made at one point in time, while accounting for possible recourse actions after all uncertainty has been resolved. There are many situations where one is faced with problems where decisions should be made sequentially at certain periods of time based on information available at each time period. Such multi-stage stochastic programming problems can be viewed as an extension of two-stage programming to a multi-stage setting. Another model extension, similarly to RO, are *probabilistic* (also called *chance*) constraints [Charnes-Cooper-Symonds, 1958], Miller and Wagner [Miller-Wagner, 1965] and Prékopa [Prékopa, 1970].

The CERBERO framework tool Architecture Optimization Workbench (AOW) [Broodney12, Broodney14] is being extended within CERBERO to be able to deal with modelling of uncertainties. More details on CERBERO approach and AOW will be discussed in Section 4.3

3.3 Model-based design space exploration

Design space exploration (DSE) is the process of searching through different system design alternatives. The aim of this process is usually to find some design which outperforms the other alternatives. Ideally one would want to find the optimal system design, but this is not always feasible. The power to operate on the space of potential design candidates renders DSE useful for many engineering tasks, including rapid prototyping, optimization, and system integration. The main challenge in DSE arises from the sheer size of the design space that must be explored.

The most challenging problems in design space exploration are managing the solution space size and using a cost function which accurately describes what the desired performance is of the system. But such challenge becomes even more complex if there is uncertainty when modelling the system and its environment. Dealing with such complex-in-nature space exploration problems is one of the targets of CERBERO. This section quickly overviews the two main approaches to this problem.

One of the natural options to perform DSE is to formulate DSE process as mathematical optimization problem, where one should optimize (minimize or maximize) system KPIs (objectives) which are functions of design decisions (decision variables) subject to possible design options and/or topologies and/or constraints on system KPIs (constraints).

For DSE purpose, one can separate modern optimization tools in two major categories**:** **operations research**-oriented (OR) tools (such as Cplex Studio [Cplex] and Gurobi [Gurobi]), and **system engineering**-oriented (SE) tools. OR tools are capable to solve very large problems with a huge numbers of decision variables and constraints. But this approach requires the domain expert to transform the system model into one of the classes of the mathematical optimization problems that are supported by a optimization solver

As an alternative, the main SE tools are designed for systems engineers to incorporate models and perform DSE using a black box / simulation-based optimization. Examples of such tools are [ModeFrontier], [ModelCenter], [Isight], [OptiY], [Nexus], [Kimeme], [Pacelab Suite], [Pacelab SysArc], [HEEDS MDO], [IOSO], and [Optimus]. These tools cannot optimize complex systems with large number of parts and/or big numbers of possible topologies by following reasons:

1. Simulation-based techniques typically require long runtime in case of multiply choices. Statistical methods offered by software can't be useful to reduce number of possible solution because of combinatorial nature of problem.
2. Usage of heuristic algorithms decreases the quality of the obtained solution.

3. Constraints are checked sequentially. Therefore, it could be hard to find a feasible solution, and the first feasible solution could be far from optimum.

Architecture Optimization Workbench (AOW) [Broodney et al 2012, 2014], [Masin et al 2013, 2014] is a single exclusion from this list that utilizes advantages from both categories of tools. ***AOW is further developed within CERBERO and uses a unique combination of modelling approach, sound software engineering and state-of-the-art mixed integer linear optimization technology.*** *Thus, it is the only existing tool that allows multi-objective optimization of system's architecture topology using the strongest existing solvers, such as Cplex [CPLEX]. Using AOW, engineers have the ability to evaluate hundreds to millions of potential architecture configurations in a matter of hours and to be able to support the architectural decisions with quantifiable benefits in driving cost and performance benefits for the program. More details on CERBERO approach and AOW will be discussed in Section 4.3*

---

4 The CERBERO approach for modelling Cyber-Physical Systems

This section presents the innovations in modelling actually in research within the CERBERO project. We organize the sections in the following using the same structure as they were reviewed in the state-of-the-art (Section 3).

4.1 CERBERO novelties on modelling complex systems

4.1.1 Models of Architecture – CERBERO's approach to cross-layer modelling

As demonstrated in this document, the practice of representing digital signal processing applications with formal MoC is currently growing, fostered by new system-level objectives such as cyber-physical entanglement or autonomic computing.

Formal MoCs are used to study application properties (liveness, schedulability, parallelism...) at a high level, often before implementation details are known. Formal MoCs also serve as an input for DSE that evaluates the consequences of software and hardware decisions on the final system. The development of formal MoCs is fostered by the design of increasingly complex applications requiring early estimates on the -functional behavior of the system under test.

On the architectural side of system development, heterogeneous platforms are becoming ever more complex. Languages and models exist to formalize performance-related information of a hardware system. They most of the time represent the topology of the system in terms of interconnected components and focus on time performance. However, the body of work on Models of Architecture (MoAs) is much more limited and less neatly delineated than the one on MoCs [PELCAT18].

The use of a couple MoC-MoA for DSE is illustrated in the following figure representing a Y-chart where an application is modelled using a MoC, an architecture is modelled using an MoA and application/architecture are redesigned based on early efficiency metrics (representing the defined/desired KPIs) extracted from a simulation phase.

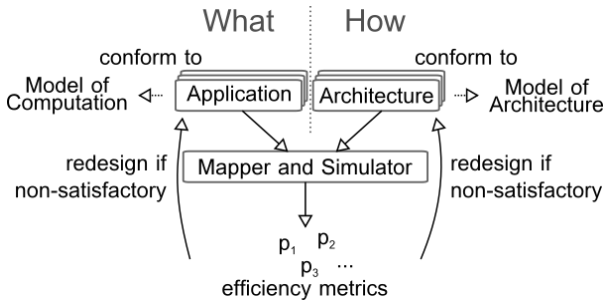


Figure 6: Design Space Exploration based on the MoC-MoA couple.

.

For self-adaptation purposes, a couple MoC-MoA is also an asset, enabling system self-scheduling, measured KPI interpretation as well as multi-system management.

Consequently, couples MoC-MoA help crossing barriers between design layers and provide low-complexity CPS representations at system level.

As part of the CERBERO project, we are proposing a definition [PELCAT17] for the concept of an MoA that recognizes the importance of MoAs in the process of system design.

Model of Architecture Definition [PELCAT17]:

> A Model of Architecture (MoA) is an abstract efficiency model of a system architecture that provides a unique, reproducible cost computation, unequivocally assessing an architecture efficiency cost when supporting the activity of an application described with a specified MoC.

While the reproducible cost computation prevents mere block decompositions from being considered an MoA, abstraction makes it possible to reuse a unique MoA for several KPIs.

Activity Definition [PELCAT17]:

> Application activity corresponds to the amount of processing and communication necessary for accomplishing the requirements of the considered application during the considered time slot. Application activity can take different shapes and is composed of abstract processing and communication tokens.

As an example of a simple MoA, the Linear System-Level Architecture Model (LSLA) has proven efficient in modelling the power consumption of a heterogeneous multi-ARM system [PELCAT17]. LSLA is representing an additive KPI whose amount depends on both computation and communication amounts. The next figure represents a model conforming to the LSLA MoA. This model has been learnt automatically from energy measurements and reveals that processing elements (PE) on the left consume about 230mW when running parts of the test application and PEs on the right consume about 1.2W when running parts of the test application. More details on this example are given in [PELCAT17].

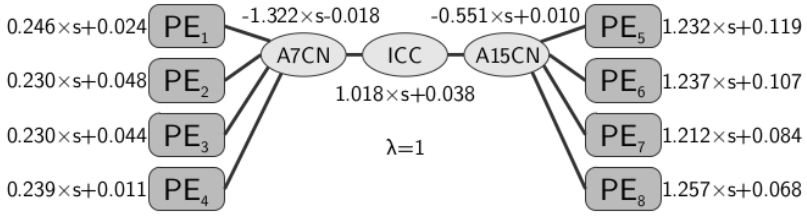


Figure 7: Example of an architecture model conforming to the LSLA MoA.

This topic is closely related to the research on KPIs discussed in Section 4.1.4.

4.1.2 System level multi-view modelling

The CERBERO modelling methodology (and the tool framework) conceives the model of a CPS system as a collection of many interdependent, simpler models, each of which abstracts and expresses a viewpoint over the same system-under-design. This way of composing a complex system model out of many different views (or aspects) is called multi-view modelling, and its basic concepts were discussed already in section 3.1.2. The support and research topics for multi-view modelling in CERBERO are specially targeted in the DynAA design tool, by TNO [OLIVEIRA13]; the AOW optimization tool, by IBM; and in the CERBERO intermediate format. We discuss the DynAA tool in this section. AOW is discussed in Section 4.3, and the intermediate format in Section 4.1.3.

Within the framework tool DynAA, CERBERO investigates how multiple viewpoint models (views) can be integrated and combined for producing system analysis results and perform system-level simulation. DynAA works with four fundamental viewpoints to describe a CPS:

- the **behavioral model,** which describes the functional composition of a task and its execution sequence;
- the **task model**, which captures the parallelism and the event handling;
- the **physical model**, which describes the hardware configuration of the implementation.
- the **mapping model**, which describes a binding between the task and the physical models.

The behavioral model (or viewpoint) uses similar semantics as a UML activity diagram [OMG07], by specifying a sequence of operations, called *control flows*. Unlike UML activity diagrams, the DynAA behavioral model does not support fork ()/join () and barrier constructs, as these constructs are associated with modelling (dynamic) concurrency and covered by the task model. In other words, the behavioral model only captures purely sequential behavior inside a task. Figure 8 depicts an example of a behavioral diagram in the DynAA modelling language. The behavioral aspect supports different types of operations, such as processing operations, communication operations (either send or receive) and the delay operations. Operations are annotated with computational load information, such as number of integer operations needed, number of floating point operations needed, size of communicated messages, etc.

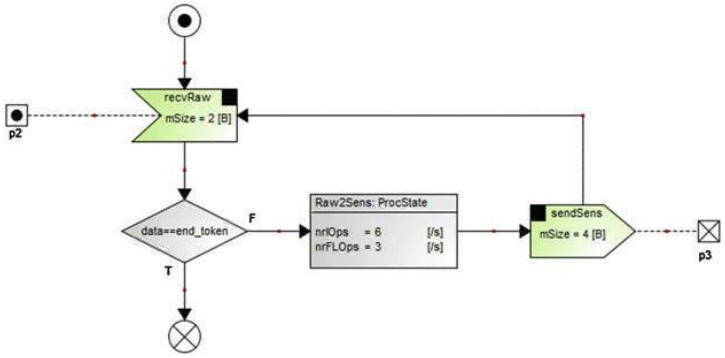


Figure 8: Behavioral model (viewpoint) in DynAA

The functional blocks of the behavioral model are grouped into tasks, i.e. parallel executable units of code based on the external event handling and concurrency (real-time) concerns. Hence, **composition is the first way we use to combine modelling views**. The task model (with the associated behavioral model) captures the programmatic properties of the design. Note that no hardware and physical communication related properties are incorporated in the model. Tasks coordinate the work by communicating and synchronizing with each other, i.e. tasks are interconnected. The connections are called links and have flow semantics (Kahn process networks MoC, more details on that at [CERBERO_D3.5]). Figure 9 shows the task model example – written using the DynAA graphical language – for a very simple application.

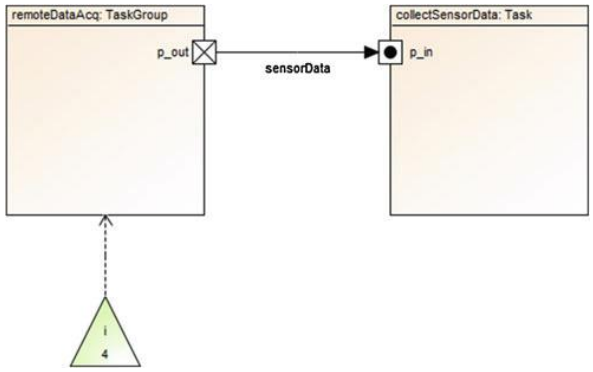


Figure 9: Task model (viewpoint) in DynAA

Tasks inherit resource requirements from the blocks in the associated behavioral model. Consequently, **a task can have a specified memory footprint, a computation load, and a set of required hardware resources (a list of device type names), based on the exchange of information with another modelling viewpoint**.

The physical model (viewpoint) describes an abstraction of the physical resources of the system being modelled. It models the hardware resources that are used to implement/run an application. Hardware resources include processing resources (processors, cores), communication resources (communication interfaces, communication networks), storage resources (memory) and energy resources (power supply, battery). Hardware resources can be shared (processor, memory, network) or can be consumed and replenished (energy).

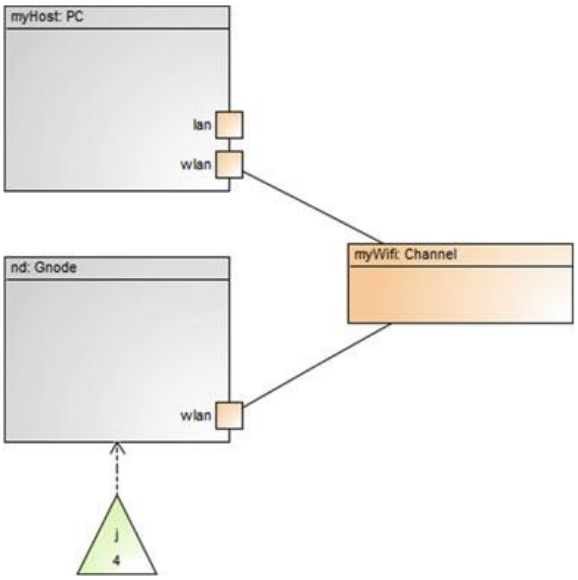


Figure 10: Physical model (viewpoint) in DynAA

Hardware related characteristics start playing a role when the task graph is mapped to the physical model: the task network is executed on a physical hardware configuration consisting of processing nodes connected by communication links (lower-right graph). So, for example the memory footprint value is used to check if a node can accommodate that task in its memory. The computational load is used to derive overall calculation time of a task mapped onto a node. The mapping relationship between the task model and the physical model is very important and may be very complex. For this reason, there is an extra view – the mapping model – whose purpose is only to integrate the concepts of the other two viewpoints. **Integration models (viewpoints) are another way CERBERO uses to operationalize multi-view modelling.** The DynAA tools can combine the information of these multiple viewpoints on the same CPSs to generate simulation code. The simulation code is used for design space exploration (see Sections 3.3 and 4.3) and deriving system KPIs (see Sections 3.1.4 and 4.1.4).

In summary, the CERBERO project proposes new techniques on the operationalization of multi-view modelling for purposes of system analysis and system simulation. These techniques are based on:

- **Composition of modelling views** – propagation of component properties throughout different views;
- **Combination of modelling views –** component properties our of different views, are combined to generate a system property, e.g. load of a task is combined with the processing capability of a node to derive estimates for the execution time of the task.
- **Integration viewpoints –** when the interaction between viewpoints is complex, we use an extra viewpoint (e.g. the mapping model) to specify the interoperability between viewpoints.

4.1.3 The CERBERO intermediate format : sharing models for increased tool interoperability

The CERBERO approach for cross-layer (Section 4.1) **and multi-view** (Section 4.1.2) **model-based design requires an efficient sharing of system models between design tools**. Tool interoperability – that is, the efficient sharing of model information between different tools – is a major challenge in the design community. As discussed in Section 3.1.3, the current major approach to this task is to use metamodels to guide the information sharing. But this approach does not solve neither the multi-view interoperability problem, nor the multi-tool interoperability problem, nor the model maintenance problem.

CERBERO innovates by proposing a two-layered intermediate format for sharing model information between tools (and model views). The CERBERO Intermediate Format (CIF) strives for:

- **sharing system model information across different levels of abstraction and different modelling views**. The modelling of CPSs is intrinsically multi-disciplinary, multi-view, and involves different abstraction layers. Any unique model representation for the whole system that cannot cope with these intrinsic characteristics is doomed to fail. The model information should be equally adequate and accessible for the representation of several views, to the different tools manipulating the model (modelling, analysis, code-generation, runtime, validation), and for manipulation at different abstraction levels. In other words, an intermediate format that fully exploits the idea of a one-model-with-multiple-views on the system model.
- **enabling different tools to access information about a system model with minimally incorporating details of the metamodels used in other tools**. When a tool looks at the system information in the intermediate format, it sees ... only information! Tools should be able to read, understand, and manipulate the model data with minimal knowledge on how this data is organized in other tools.

To solve tool interoperability problems, CERBERO proposes an innovative, two-layered intermediate format that detaches the model data from the metamodel (schema) structure information – see Figure 11. The first layer of CIF is called the ***instance layer*** and represents the existence of *things with properties*, independent of any classes, schema, or pre-classification to which these things may belong. The second layer of CIF is called ***classes layer*** and consists of definitions for *classes* and *metamodels (schemas)*. Classes of interest are then defined by sets of properties. The CIF approach is largely based on the work of Parsons and Wand [PARSONS] and follows the most modern development in the information representation and database community: enables semantic operability instead of structure interoperability, builds on non-schema databases and non-schema information modelling.

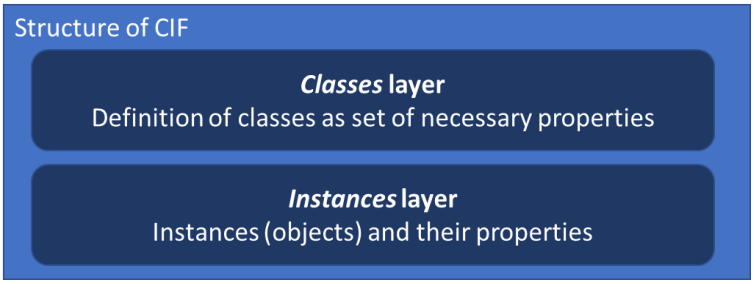


Figure 11: Layers of the CERBERO Intermediate Format

We use an example to explain the innovations introduced by the CERBERO Intermediate Format based on the simple system described in Figure 12. The system model describes two tasks (Sender and Receiver), whose ports (tx, rx) are connected by a communication link. The metamodel to describe such models is part of the CERBERO tool DynAA and can be seen in Figure 5, in Section 3.1.3.

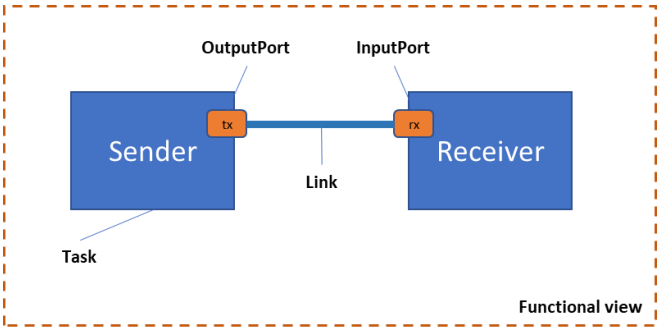


Figure 12: Example of a model

Classical metamodelling based approaches to interoperation between modelling tools use the metamodel to derive an intermediate format for storing and sharing the information of this model. An example could be an XML file as the snippet in Figure 13, which represents the sender-to-receiver model. Notice that the nodes in this XML file reflect the types and properties in the metamodel (task, outport, name, etc.). That implies that for another tool to read this format, it must understand the metamodelling structure of the tool that wrote the file. We say in this case, the metamodel information is part of the modelling information.

```
<?xml version="1.0" encoding="UTF-8"?>
<functionalModel>
    <task>
        <name>
            sender
        </name>
        <outport>
            <name>
                tx
            </name>
        </outport>
    </task>
    <task>
        <name>
            receiver
        </name>
```

Figure 13: Snippet of an XML file representing the model in Figure 11

In the CERBERO Intermediate Format, schema structure (types) are separated from entities and properties in two different layers. The idea for the CIF is depicted in Figure 10, where elements in the *instance layer* are represented with blue dots, and elements in

the *classes layer* are represented with yellow dots. Each element in the instance layer is an *instance(thing)* with properties**.** Properties can be simple, such as the name of a task (e.g. **name:** task#sender); or mutual, such as the relationship between a link and a source port (e.g. **from**). Note that mutual properties are properties shared by two instances. The *instances* have not yet any classification. Classes are defined by the set of properties an instance should have and are declared in the *class layer*. For example, a **link** is any instance containing the property **from** and the property **to**.

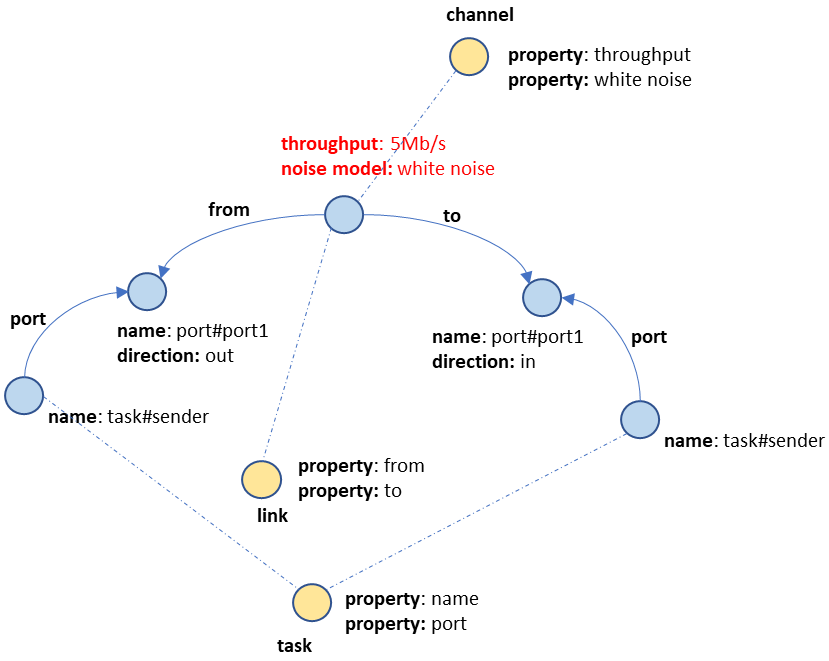


Figure 10: Representation of the sender-receiver model as concepted in the CERBERO Intermediate Format

The use of such a two-layered intermediate format can help solving many interoperability problems. For example (see Figure 10), suppose a second tool models links between tasks as communication channels – called **channel** – in their internal metamodel. Also, this tool attributes **throughput** and **noise model** to the channel to make its analysis possible. Such information – as well as the new classification – can be attributed to the same *instance* node without any prejudice or modification for the other tool. It can also be performed without knowledge of other tools.

The conception and implementation of the CERBERO Intermediate format is still a work on-going and CERBERO is still experimenting with different use cases that demonstrate the adequacy of this technique to share modelling information between tools.

In summary, the CERBERO project innovates with a different way to store and share model information and metamodels between tools – the CERBERO Intermediate Format. The principles of CIF are:

- Separation of model content information from model structure/schema information

- Enable modelling tools to read/enrich a model information without (or with minimal) knowledge on the other tools schema/metamodels.

4.1.4 Modelling Key Performance Indicators

A KPI is a quantitative, relevant measure – as for example total power consumption, reliability, system availability, computational performance, etc. – that can be calculated/evaluated over a given system model. The use of KPIs is a common practice in the design evaluation and DSE of CPSs (for a grounding discussion see Section 3.1.4). **Despite its importance, designers mainly define KPIs in an ad-hoc manner, and there is no extensive work on formalizing KPIs as well defined and structured models.** Such lack of formalism hinders much of the potential that KPIs could offer to a design process – such as automating parts of the evaluation and analytically understanding the properties of a KPI. CERBERO intends to change that following a seminal approach introduced recently in [MASIN ET AL., 2013].

The evaluation of any KPI out of its system model implies that designers must determine a way to calculate them. And such calculation mostly can be expressed by means of a certain mathematical structure, or as we call an ***algebra***. For example, the total energy consumption of a machine(system) is often evaluated by summing up (synchronously) the energy consumed by each of its integrating components(sub-systems). Such operation could be expressed mathematically as:

$$E_{total}(t) = \sum_{i \in S} E_i(t),$$

where *S* is the set of components of the machine. Notice that this calculation defines a certain mathematical structure: it is a summation over a property present (or measurable) in each element of a set.

Now, consider as well the total monetary cost of a machine(system). Such KPI is often expressed as:

$$C_{total}(t) = \sum_{i \in S} C_i(t),$$

and as such it presents the same mathematical structure as the energy consumption defined in the first example. One could argue that both KPIs could be calculated by following the same set of operations but applied on a different set of components of the system and/or a different set of their properties. It turns out that many other KPIs, in different systems, are often defined with the same underlying structure. Take for example the total weight of a system, the total availability time (of sequential services), total volume of accumulated liquid (in a system of hydraulic tanks), etc.

Like the KPI family exemplified above, it is possible to identify many others that are recurrently used when designing complex systems. Examples of other common families of KPIs are:

- The KPI is a maximum value within a set of properties (values);
- The KPI is a minimum value within a set of properties (values);
- The KPI is an average value within a set of properties (values);
- The KPI is a weighted sum over a set of component's properties;

- The KPI is a weighted sum over paths in a network graph – e.g. network throughput.

KPIs can be much more complex than the summation algebra presented as example above. Some will require considering relationships between components, network topology, and even dynamic aspects (dependent on time). Despite its complexity, the calculation of a KPI falls always into some definable *algebra*, and often exhibiting a well know calculation structure. CERBERO uses this fact to model KPIs in a more formal way. In other words, **CERBERO proposes ways to formally model and classify KPIs according to the mathematical structure they exhibit (or need) in their calculations**. This methodology is extensively discussed in the deliverable [CERBERO_D3.4].

There are many advantages in describing KPIs in a formal way:

- It **enables automated KPI evaluation** for many KPI families, where the designer just needs to specify part of the data set in the system model over which the evaluation should happen.
- **Analysis** over a certain KPI family already exposes some properties of the system, even when no KPI evaluation is done. For example, additive KPIs such as the ones discussed previously cannot decrease in value when new components are added to the system (given that there can be no negative property value).
- **Complex KPI calculations** can be defined based formal mathematical methods, e.g. process algebras. Moreover, the description of KPI families are not limited to be done by using mathematical expressions or set of equations. It can also be formally described as a set of procedures, algorithms, combination of other KPIs, etc.
- **Improved DSE** can be achieved for certain KPI families due to their mathematical properties. For example, some KPI families can be strictly linear, or monotonic, or continuous-by-parts, etc.
- Combining **formal models of KPIs with the CERBERO intermediate format** (see Section 4.1.3) yields a powerful KPI evaluation tool. The *algebras* defined for many usable KPI families map directly into traversal operations over property graphs such the one used to implement CIF. This combination further eases automation of the KPI evaluation.
- Last, but not least, **a formal approach to KPIs eases the change of KPIs in adaptive processes**. During system adaptation, KPIs are often calculated to decide how the system should change. But the KPIs used themselves may change depending on what is the actual system state and how the system wants to adapt (dynamic system goal).

The CERBERO methodology to model and analyze KPI is discussed in detail in the CERBERO deliverable D3.4 [CERBERO_D3.4].

4.2 CERBERO novelties on modelling for reconfiguration and self-adaptation

4.2.1 CERBERO novelties on modelling concurrent and distributed behavior

The CERBERO project has for objective to demonstrate the capacity of formal MoCs to efficiently drive CPS design and self-adaptation. MoCs can capture essential properties of an application that enable expressing concurrency, causality, event dependency, memory locality, etc. And this regardless of the system architecture.
Separation between application (MoC) and architectural (MoA) concerns should not be confused with software (SW) / hardware (HW) separation of concerns. The software/hardware separation of concerns is often put forward in the term HW/SW co-design [HA17]. Software and its languages are not necessarily architecture-agnostic representations of an application and may integrate architecture-oriented features if the performance is at stake. This is shown for instance by the differences existing between the C++ and CUDA languages. While C++ builds an imperative, object-oriented code for a processor with a rather centralized instruction decoding and execution, CUDA is tailored to GPGPUs with a large set of cores. As a rule of thumb, software qualifies what may be reconfigured in a system while hardware qualifies the static part of the system.
This separation is currently blurred by new hardware paradigms such as Dynamic and Partial Reconfiguration (DPR) and Coarse Grain Reconfigurable substrates (CGR), paradigms largely exploited within the CERBERO project.
By using a couple MoC – MoA, the CERBERO project provides new features to the system designer such as:

- Self-adaptation with runtime scheduling benefiting from internal knowledge of application concurrency and architecture parallelism,
- Application-awareness with application parameters crossing the boundary between the applicative layer and the system management layer,
- Hardware reconfiguration at two different levels of granularity (CGR and DPR),
- Software reconfiguration over multiple processing elements with shared or distributed memory,
- Full data-driven execution where processing elements and sub-systems can be asynchronous and processing is triggered by the arrival of data.

The choice of these systems features is motivated by the recent evolutions of CPS embedded platforms. Platforms such as the Xilinx Zynq Ultrascale+, Qualcomm Snapdragon 845, Intel Arria 10 SoC or Samsung Exynos 9 all integrate a variety of processing elements, often exposing reconfigurable hardware or hardware acceleration IPs.

The CERBERO exploited MoCs are detailed in Deliverable D3.5 [CERBERO_D3.5].

4.2.2 CERBERO novelties on modelling of uncertainty

Concurrency modelled by flexible MoC, such as PiSDF, can be combined with uncertainty modelling techniques for better evaluation, analysis, and, consequentially, design of

reconfigurable and self-adaptive CPS that should address different operational conditions, both in internal status and external environment.

The CERBERO exploited uncertainty modeling is detailed in Deliverable D4.4 [CERBERO_D4.4].

4.3 Model based design space exploration in CERBERO

As previous work to CERBERO, a team from IBM Research lab in Haifa, Israel has developed an approach that promises to bestow the power of optimization onto Systems Engineers [Broodney et al., 2012]. Such effort was realized in a software tool called AOW, which is further extended within CERBERO.

Using AOW, the engineer can model the composition rules (a.k.a. architectural pattern, template) of the required system. The functional requirements are modelled, including the relations (data flow, energy flow, etc.) between them, and potential mappings to the physical components are specified. The physical structure of the system represents the composition rules. In the example in Figure 14, a Power Distribution System basic structure is depicted. The block PDB represents a multitude of such blocks in a final architecture and the connections in the diagram represent the structural requirements and constraints for a correctly constructed system. This modelling approach is called "Concise Modelling".

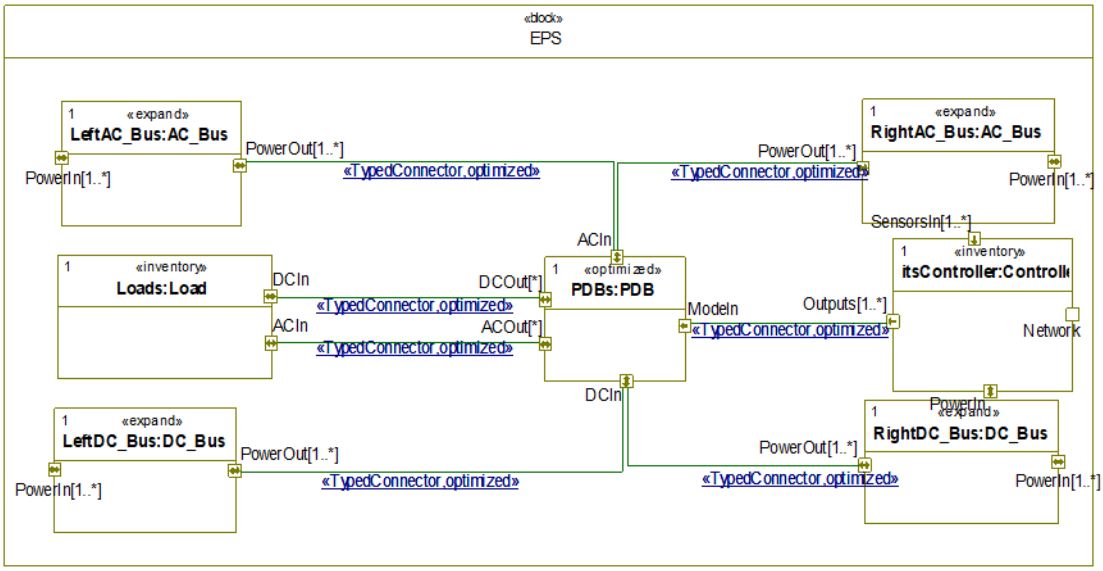


Figure 14: Example of a SysML diagram for design space exploration in CERBERO tool AOW. The systems example is based on a power distribution plant

Optimization goals are specified in a special "objective" blocks. There is a possibility to add constraints (textual) blocks as well. Systems Engineers can develop libraries of reusable metrics in extended SysML Parametric Diagrams [Masin et al., 2013 and 2014]. In CERBERO, this module will be coupled with the new KPI models.

All the inputs above are used by the software to generate mathematical optimization program in OPL language, to be run in the IBM ILog CPlex Optimization tool, which is the market leading Linear Programming solver. Since the goals are multiple and usually

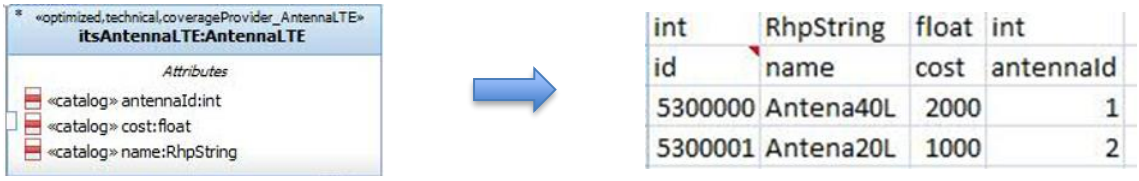
conflicting, the solution will propose several solutions along the efficient frontier, meaning that for neither solution no goal can be improved without adversely affecting another. Patent pending algorithms assure that the options provided are as diverse as possible, thus offering the engineer a fresh, previously unseen, aspect on the architecture of the system. The results of the optimization are back-annotated into the SysML tool for the engineer to view.

The AOW modelling language (used in the tool) is built on top of standard SysML and allows definition of different AOW concepts applying stereotypes to SysML elements.

These concepts include:

- viewpoints,
- mappings between viewpoints,
- integration features,
- optimization concepts such as constraints, goals and decision variables,
- metric libraries, i.e. model independent metric definitions that could be applied top AOW model introducing set of constraints on potential system architecture,
- mappings between model elements and library metrics.

Viewpoints representing different levels of abstraction or different system concepts. AOW currently supporting <functional>, <technical>, <geometrical> and <reliability> viewpoints. Affiliation of the element to the specific viewpoint expressed by assigning to this element corresponding stereotype. Mappings between different viewpoints expressing by adding dependencies between corresponding parts. These dependencies should have stereotype <mappedTo> (for mapping between functional and technical viewpoints) and <allocatedTo> (for mapping between technical and geometrical viewpoints). Integration features includes <catalog> stereotype applying to SysML block, that means that parts of this block are tacking from some catalog represented by corresponding Excel table.



Another integration feature is an <inventory> stereotype, which is applied to SysML part or link (dependency, flow or connector) indicates that such part or link represents a set of parts or links given by the Excel table. Optimization constraints expressed by SysML constraints with special stereotypes and multiplicities on parts and link ends. Decision variables expressed by <optimized> stereotype that can be applied to the part or link indicating that total number of these parts or links and actual type of these parts or links (in case of <catalog> blocks) are determined by optimization, or, if applied to attribute indicates that actual value of this attribute determined by optimization. Optimization goals expressed by stereotype applied to <optimized> attribute, where value of such attribute is equal to some metric or subject to some equality constraint.

Metric libraries include different types of metrics and expressed by special type of parametric diagrams or by special textual language. One can distinguish 3 different types of metrics. The first type determines a new sets, parameters and decision variables that created during transformation of SysML model to optimization model. The second type

determines calculation of compound parameters that performed before the optimization process start. The last type determines optimization constraints that defined by some compound system metrics (such as energy flow for example). To apply a metric to the model one should apply special stereotype to SysML part, link or attribute (for textual metrics) or use AOW mapping UI (for diagram metrics).

Process of translation of AOW model to mathematical optimization model consist of several steps. At the first step SysML model as well as Excel tables converted to CIF, the CERBERO intermediate format. Next, models in CIF are going through a series of transformations that combine Excel and SysML model, combine different viewpoints according to rules defined in metric library and build option graph that represent all possible system architectures. Afterwards, additional sets of elements are building according to corresponding metric rules. It is important to note that parts having attributes with same names combined into attribute sets. Attribute set represent classification by property concept and can be used in metrics and SysML constraints. At the next step calculation of the pre-optimization metrics performed. These defines all optimization parameters. Finally, parts of the model in CIF are translated to OPL programming language. These includes generation of decision variables, optimization constraints and goals.

CERBERO modelling methodology will extend AOW modelling paradigm in a several directions:

1. The set of AOW viewpoints and mappings will modified and extended to allow modelling of CPS aspects in straightforward and concise way.
2. Modelling language will extend to provide capabilities that allow definition of various aspects related to system and environment uncertainty. Namely, this will include ability to define: probability distributions, scenarios, uncertainty sources, system properties affected by these sources, chance constraints, adjustable and non-adjustable decision. CERBERO methodology will also enable translation of uncertain system and environment aspects mentioned above into computationally tractable mathematical optimization problems utilizing state of the art robust and stochastic optimization methods. As a starting point for developing such methodology we consider [Shindin et al, 2014]
3. CERBERO methodology will extend AOW modelling language to enable definition of continuous dynamic system aspects, including related decision variables, constraints and objectives for mathematical optimization problems. This will also include automatic translation of corresponding continuous aspects of the system into state of the art CLP problems that can be solved by efficient algorithms.

5 Implementing the CERBERO modelling approach

In the CERBERO project, research for the modelling methodology flows directly into the tools: research in each innovative topic is adopted by at least one of the tools in the CERBERO framework. By adopting the new modelling technique into a tool allows to immediately address issues such as the feasibility and automation of the technique in a modelling environment.

Moreover, the validation of each modelling aspect is re-enforced using the tool in use cases. During the presentation of results for each use case, CERBERO will also include the validation results over using a certain modelling technique or innovations discussed in Section 4.

In Table 3, we summarize per modelling innovation discussed in this document (Section 4), the tools where the modelling technique is absorbed and operationalized **and** the use case where the modelling aspect will be validated. All the topics mentioned in the table are mature in their research and the incorporation of them in the respective tool is already started. It must be noticed as well that there is a noticeable predominance from the tools AOW and DynAA in the table. This is clear to understand: these two tools are based around a complete modelling framework, including modelling language and GUI. For example, AOW uses SysML and uses IBM Rational Rose as graphical front-end. Other tools' primary focus is mainly on manipulating the model – analysis, code generation, verification, etc. – and as such, only the modelling techniques related to their use are incorporated. Further, each tool is also coded with a color to help in their quick identification.

Table 3: Per modelling aspect in CERBERO, the tools where they are incorporated and the use cases where they are validated.

	Space Exploration	Smart Travelling	Ocean Monitoring
Modelling complex systems			
Models of Architecture	**SPIDER** Uses MoAs to quickly evaluate online options for the deployment of an application (HW/SW partitioning).		
Multi-view modelling	**SPIDER** Uses combinations of hardware and software	**DynAA** System simulation fully built on combinations of multi-view models: functional,	

	model viewpoints.	concurrency, physical, network, and mapping.	
		AOW Optimization combining functional, physical, and constraints view.	
Tool interoperability via CERBERO intermediate format		**DynAA** Exchange of model information with AOW.	
	PREESM (Intended work) Exchange of model information with Spyder and Gaph.	**AOW** Exchange of model information with DynAA.	
	CIF – CERBERO Intermediate Format library This tool is an external library and API especially created within the CERBERO project to facilitate the adoption of the proposed intermediate format by other tools. This API is now used by DynAA and AOW (see above), but its use by other tools in the CERBERO framework is planned for the second half of the project.		
Modelling of Key Performance Indicators	**SPIDER** KPIs for execution time, latency, energy consumption	**DynAA** KPIs for system response time (to use), battery lifetime, user satisfaction	
	PREESM KPIs for energy consumption, throughput, chip area, reconfiguration time	**AOW** KPIs for (monetary) cost, network communication and energy	

	Artico3 KPIs for energy consumption, throughput, chip area, reconfiguration time	
Modelling concurrent systems		
Models of Computation	**SPIDER** Demonstrates usage of Synchronous data flow and Bulk Synchronous models	**MECA** Demonstrates the Situated Cognitive Engineering MoC
	PREESM Demonstrates usage of PiSDF models of computation	**DynAA** Demonstrates uses of discrete events, petri nets, and process networks models
	Artico3 and MDC Demonstrates register transfer and synchronous data flow models of computation	
Design Space Exploration		
Advanced Modelling Techniques for Design Space Exploration		**AOW** Incorporates advanced optimizers that can deal with uncertainty in the modelled environment. Incorporates also advanced

		modelling techniques to describe combinatorial problems as hybrid fluid-dynamics models – this eases the optimization algorithm.	
		DynAA Demonstrates simulation-based design space exploration. Modelling for simulation purposes include behavioral and descriptive semantics.	

6 References

[ADEL09]	Adela del-Rio-Ortega, Manuel Resinas; *Towards Modelling and Tracing Key Performance Indicators in Business Processes*, Actas de los Talleres de las Jornadas de Ingeniería del Software y Bases de Datos, Vol. 3, 2009
[ANDERSON - NASH, 1987]	Anderson, E.J., Nash, P. Linear Programming in Infinite Dimensional Spaces. Wiley-Interscience, Chichester, 1987
[ANDERSON, 1978]	Anderson, E.J. A Continuous Model for Job-Shop Scheduling. Ph.D. Thesis, University of Cambridge, Cambridge, 1978
[ANDERSON, 1981]	Anderson, E.J. A new continuous model for job-shop scheduling. *Int. J. Syst. Sci*. 12, 1469–1475, 1981
[BEALE, 1955]	E. M. L. Beale. On Minimizing A Convex Function Subject to Linear Inequalities. *Journal of the Royal Statistical Society. Series B (Methodological)* Vol. 17, No. 2 (1955), pp. 173-184
[BELLMAN, 1953]	Bellman, R.: Bottleneck problems and dynamic programming. Proc. Natl. Acad. Sci. 39, 947–951, 1953
[BEN-TAL – EL GHAOUI – NEMIROVSKI, 2009]	Ben-Tal A., El Ghaoui, L. and Nemirovski, A. Robust Optimization. *Princeton Series in Applied Mathematics*, Princeton University Press, 2009.
[BERTSIMAS ET AL, 2015]	Bertsimas, D., Nasrabadi, E., Paschalidis, I. C. (2015). Robust Fluid Processing Networks, IEEE Transactions on Automatic Control, 60 (3), pp. 715 – 728.
[BIRGE-LOUVEAUX,1997]	Birge, J. R. and Louveaux, F. Introduction to Stochastic Programming. *Springer Series in Operations Research*, Springer-Verlag, New York, NY, 1997.
[BROODNEY ET AL., 2012]	Broodney, H., Dotan, D., Greenberg, L., and M. Masin, 2012, "Generic Approach for Systems Design Optimization in MBSE", *INCOSE 2012*.
[BROODNEY ET AL., 2014]	Broodney, H., Masin M., Shany, U., Shindin, E., Kalawsky, R., Joannou, D., Tian, T., and Sanduka, I., "Leveraging Domain Expertise in Architectural Exploration", *CSDM 2014*.
[CAFE13]	Daniel Chaves Cafe, Filipe Vinci dos Santos, Cecile Hardebolle, Christophe Jacuet, and Frederic Boulanger. *Multi-paradigm semantics for simulating SysML models using SystemC-AMS*. In FDL 2013. IEEE, 2013
[CANC15]	Daniela Cancila, Hadi Zaatiti, Roberto Passerone. *Cyber-Physical System and Contract-Based Design: A Three Dimensional View*. In WESE, 2015
[CERBERO17]	CERBERO EU Project, Online: http://www.cerbero-h2020.eu
[CERBERO_D2.7]	CERBERO EU Project, *Deliverable D2.7 – CERBERO Requirements*, Online: http://www.cerbero-h2020.eu

[CERBERO_D3.4]	CERBERO EU Project, *Deliverable D3.4 – Key Performance Indicators*, Online
[CERBERO_D3.5]	CERBERO EU Project, *Deliverable D3.5 – Models of Computations*, Online: http://www.cerbero-h2020.eu
[CHANG97]	W. T Chang, S. Ha, and E. A Lee. *Heterogeneous simulation - mixing discrete-event models with dataflow*. The Journal of VLSI Signal Processing, 15(1):127–144, 1997.
[CHARNES-COOPER-SYMONDS,1958]	A. Charnes, W. W. Cooper, G. H. Symonds. Cost Horizons and Certainty Equivalents: An Approach to Stochastic Programming of Heating Oil. *Management Science - Management* , vol. 4, no. 3, pp. 235-263, 1958
[CPLEX]	IBM ILOG CPLEX Optimization Studio, http://www-01.ibm.com/software/integration/optimization/cplex-optimization-studio
[DANTZIG, 1955]	G. B. Dantzig. Linear Programming Under Uncertainty. *Management science*, 1:197-206, 1955
[EMF]	Eclipse.org, *Eclipse Modelling Framework*, Online: https://www.eclipse.org/modelling/emf/
[ERMOLIEV-WETS,1988]	Yu. Ermoliev and R. J.-B. Wets (eds.): Numerical Techniques for Stochastic Optimization Problems. Springer, Berlin, 1988.
[FEILER12]	Peter Feiler, David Gluch; *Model-Based Engineering with AADL: An Introduction to the SAE Architecture Language*, Addison Wesley, ISBN-13 978-0134208893, 2012
[FMI14]	FMI development group. *Functional mock-up interface for model exchange and co-simulation, 2.0*. https://www.fmi-standard.org, 2014.
[GME]	WEBGME.org; *Web GME*, Online: https://webgme.org/
[GUROBI]	http://www.gurobi.com
[HA17]	Soonhoi Ha, Jürgen Teich (Eds.), *Handbook of Hardware/Software Codesign*, Springer, 2017.
[HANEVELD-VAN DER VLERK,1999].	Klein Haneveld, W. K. and Van der Vlerk, M. H. Stochastic Integer Programming: General Models and Algorithms. *Annals of Operations Research*, 85, 39–57, 1999.
[HEEDS MDO]	http://www.redcedartech.com/products/heeds_mdo
[INTOCPS]	Integrated Toolchain for Model-based design of Cyber-physical Systems, Online: http://projects.au.dk/into-cps/
[IOSO]	http://iosotech.com/product.htm
[ISIGHT]	http://www.3ds.com/products/simulia/portfolio/isight-simulia-execution-engine/overview/
[ISOIEC]	ISO/IEC; *International standard ISO/IEC numbers 10746-1, 10746-2, 10746-3, and 10746-4*, 1996-1998

[KARSAI10] G. Karsai, F. Massacci, L. Osterweil, I. Schieferdecker; *Evolving Embedded Systems*, Computer 43(5), Vol. 34, 2010

[KERN11] Heiko Kern, Axel Hummel, Stefan Kühne; *Towards a comparative analysis of meta-metamodels*. Proceedings of the SPLASH Workshops, 2011

[KIMEME] http://www.kimeme.com/products.php

[LEE98] Lee, E. A., & Sangiovanni-Vincentelli, A. (1998). *A framework for comparing models of computation*. IEEE Transactions on computer-aided design of integrated circuits and systems, 17(12), 1217-1229.

[MASIN ET AL., 2013] Masin, M., Limonad, L., Sela, A., Boaz, D., Greenberg, L., Mashkif, N., and R. Rinat, 2013, "Pluggable Analysis Viewpoints for Design Space Exploration." *CSER 2013*

[MASIN ET AL., 2014] Masin, M., Broodney, H., Brown, C., Limonad, L., Mashkif, N. and A. Sela, 2014, "Reusable derivation of operational metrics for architectural optimization", *CSER 2014*.

[METAEDIT] MetaCase Inc.; *MetaEdit* Domain-Specific Modelling (DSM) environment*, Online: https://www.metacase.com/products.html

[MILLER-WAGNER, 1965] Miller, L.B. and Wagner, H., Chance-constrained programming with joint constraints, *Operations Research*, 13, 930-945, 1965.

[MODEFRONTIER] http://www.modefrontier.com/

[MODELCENTER] http://www.phoenix-int.com/software/phx-modelcenter.php

[MORIN09] B. Morin, O. Barais, J.M. Jezequel, F.Fleurey, A.Solberg, *Models at runtime to support dynamic adaptation*, Computer, IEEE Computer Society, 2009

[NEMIROVKSI – SHAPIRO, 2006] Nemirovksi, A., Shapiro, A., Convex Approximations of Chance Constrained Programs, *SIAM Journal of Optimization*, Vol. 17, No. 4, pp. 969–996, 2006.

[NEXUS] http://ichrome.eu/nexus/

[NORKIN-PFLUG-RUSZCZYŃSKI,1998]. Norkin, V. I., Pflug, G. C. and Ruszczyński, A. A Branch and Bound Method for Stochastic Global Optimization. *Mathematical Programming*, 83, 425–450, 1998.

[OLIVEIRA13] Julio Oliveira, Zoltan Papp, Relja Djapic, Job Oostveen; *Model-based design of self-adapting networked signal processing systems*, In Proceedings of the Self-Adaptive and Self-Organizing Systems Conference (SASO), 2013

[OMG07] O.M.Group: OMG *Unified Modelling Language superstructure Specification*, 2007

[OMG12] OMG systems modelling language, version 1.3. Technical report, OMG, 2012

[OPTIMUS] http://www.noesissolutions.com/Noesis/

[OPTIY] http://www.optiy.eu/

[PACELAB SUITE] http://www.pace.de/products/preliminary-design/pacelab-suite.html

[PACELAB SYSARC] http://www.pace.de/?id=40

[PARSONS] Jeffrey Parsons, Yair Wand; *Emancipating Instances from the Tyranny of Classes in Information Modelling*, ACM Transactions on Database Systems, Vol. 25, No. 2, 2000

[PELCAT13] Pelcat, M., Aridhi, S., Piat, J., & Nezan, J. F. (2013). *Dataflow model of computation*. In Physical Layer Multi-Core Prototyping (pp. 53-75). Springer, London.

[PELCAT17] Pelcat, M., Mercat, A., Desnos, K., Maggiani, L., Liu, Y., Heulot, J., Bhattacharyya, S. S. (2017); *Reproducible Evaluation of System Efficiency with a Model of Architecture: From Theory to Practice*. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

[PELCAT18] Maxime Pelcat. *Models of Architecture for DSP Systems. Springer. Handbook of Signal Processing Systems*, Third Edition, In press

[PRÉKOPA, 1970]. Prékopa, A., On probabilistic constrained programming, in: *Proceedings of the Princeton Symposium on Mathematical Programming*, Princeton University Press, Princeton, pp. 113-138, 1970.

[PTOL] https://ptolemy.eecs.berkeley.edu/index.htm

[PULLAN, 1993] Pullan, M.C.: An algorithm for a class of continuous linear programs. *SIAM J. Control Optim*. 31, 1558–1577, 1993

[PULLAN, 1995] Pullan, M.C.: Forms of optimal solutions for separated continuous linear programs. *SIAM J. Control Optim.* 33, 1952–1977, 1995

[PULLAN, 1996] Pullan, M.C.: A duality theory for separated continuous linear programs. *SIAM J. Control Optim.* 34, 931–965, 1996

[PULLAN, 1997] Pullan, M.C.: Existence and duality theory for separated continuous linear programs. *Math. Model. Syst.* 3, 219–245, 1997

[ROUB13] Ella Roubtsova, Vaughan Mitchell; *Modelling and Validation of KPIs*, Proceedings of the Third International Symposium on Business Modelling and Software Design, 2013

[SCHULTZ ET AL,1998] Schultz, R., Stougie, L. and Van der Vlerk, M. H. Solving Stochastic Programs with Integer Recourse by Enumeration: a Framework Using Gröbner Basis Reductions. *Mathematical Programming*, 83, 229–252, 1998.

[SEI] Carnegie Mellon University, *Software Engineering Institute – SEI Insights*, online: https://insights.sei.cmu.edu/sei_blog/2013/11/using-v-models-for-testing.html

[SEN-SHERALI, 2006] Sen, S. and Sherali, H. Decomposition with branch-and-cut approaches for two-stage stochastic mixed-integer programming. *Mathematical Programming*, 106, 203-223, 2006.

[SHAPIRO ET AL, 2003] Verweij, B., Ahmed, S., Kleywegt, A., Nemhauser, G., and Shapiro, A. The Sample Average Approximation method applied to stochastic routing problems: A computational study. *Computational Optimization and Applications*, 24(2-3), 289-333, 2003.

[SHINDIN ET AL, 2014] Shindin, E., Boni, O. and M. Masin, 2014, "Robust optimization of system design", *CSER 2014*.

[SHINDIN-WEISS, 2014] Shindin, E., Weiss, G. (2014) Symmetric Strong Duality for a Class of Continuous Linear Program with Constant Coefficients. SIAM J on Optimization, 24(3):1102-1121.

[SHINDIN-WEISS, 2015] Shindin, E., Weiss, G. (2015) Structure of Solutions for Continuous Linear Programs with Constant Coefficients SIAM J on Optimization, 25, pp. 1276-1297.

[SHINDIN-WEISS, 2017] Shindin, E., Weiss, G. (2017) A simplex-type algorithm for continuous linear programs with constant coefficients. (under revision) arXiv preprint arXiv:1705.04959

[STRE06] T. Streichert, D. Koch, C.Haubelt, J.Teich, *Modelling and design of fault-tolerant and self-adaptive reconfigurable networked embedded systems,* EURASIP Journal of Embedded Systems, Vol.1, 2006

[SYSML] Object Management Group (OMG), *Systems Modelling Language*, Online: http://www.omgsysml.org/

[UML] Object Management Group (OMG), *Unified Modelling Language*, Online: https://www.omg.org/spec/UML/About-UML/

[VAN DER VLERK, 2009] Van der Vlerk, M. H. Convex approximations for a class of mixed-integer recourse models. *Annals of Operations Research*, 2009

[VANROY09] Van Roy, P. (2009). *Programming paradigms for dummies: What every programmer should know*. New computational paradigms for computer music, 104.

[VINC12] Alberto Sangiovanni-vincentelli, Werner Damm, Roberto Passerone. *Taming Dr. Frankenstein: Contract-Based Design for Cyber-Physical Systems*. In European Journal of Control, 2012

[VLAD12] Vladimir Dimitrieski, Milan Celikovic, Valdimir Ivancevic, and Ivan Lukovic; *A Comparison of Ecore and GOPPRR through an Information System Meta Modelling Approach*, EU Conference on Modelling Foundations and Applications. 2012

[WAW15] Frank Wawrzik, William Chipman, Javier Moreno Molina, and Christoph Grimm. *Modelling and simulation of cyber-physical systems with SICYPHOS*. In DTIS, 2015

[WEISS, 2008] Weiss, G.: A simplex based algorithm to solve separated continuous linear programs. *Math. Program., Ser. A* 115, 151–198, 2008