Interoperability between heterogeneous systems in an open environment for implementation of co-simulation

Yassine Motie

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par l'Université Toulouse 3 - Paul Sabatier

Présentée et soutenue par

Yassine MOTIE

Le 30 août 2019

Interopérabilité entre dispositifs hétérogènes en environnement ouvert pour la mise en oeuvre de co-simulation

---

Ecole doctorale : **EDMITT - Ecole Doctorale Mathématiques, Informatique et Télécommunications de Toulouse**

Spécialité : **Informatique et Télécommunications**

Unité de recherche :
IRIT : Institut de Recherche en Informatique de Toulouse

Thèse dirigée par
Alexandre NKETSA et Philippe TRUILLET

Jury

M. Reinhard GERNDT, Rapporteur
M. Gregory ZACHAREWICZ, Rapporteur
M. Rashid SALEEM, Examinateur
Mme Marie-Pierre GLEIZES, Examinatrice
M. Alexandre NKETSA, Directeur de thèse
M. Philippe TRUILLET, Co-directeur de thèse

Acknowledgements

I would like to present my sincere thankfulness to my dear father and my whole family for their great role in my life and their numerous sacrifices for me.
I would like to express my gratitude to all those who helped me during my PhD study. Many people have made invaluable contributions, both directly and indirectly to my research.
My deepest gratitude goes first and foremost to Professor Alexandre NKETSA, my supervisor, for his constant encouragement and guidance. Without his consistent and illuminating instruction, this thesis could not reach its present form.
To my co-supervisor Associate Professor Philippe TRUILLET, thank you for your kindness of looking after me wholeheartedly, for teaching me to be less "good student" and more autonomous throughout this research work.
To Professor Marie-Pierre Gleizes the Manager of the neOCampus scientific and interdisciplinary operation which aims at designing a smart, innovative, sustainable campus at Toulouse III University. You provide for me a model of courage and determination.
I would like to thank Gregory Zacharewicz and Reinhard Gerndt for agreeing to read this thesis and to be rapporteurs and Rashid Saleem to be one of the examinators.
To all my collegues in neOCampus operation who accept to share their data and work and explanations in order to ease the integration of the whole and particularly to my firend Dr Elhadi Belghache who has a sense of sharing without equal.
To Professor Sahraoui Abd-El-Kader who was there at the worst of times to restore my confidence and who often suggested me new leads and articles to facilitate my research.
To all the Teams ELIPSE in IRIT and ISI in LAAS-CNRS. Your kindest let me feel the warmth of home when I am studying in Toulouse.
I also thank the members of the jury and in particular my thesis reporters
for the time given to the reading of the report and whose enlightened remarks will be useful for me.

Résumé

Le grand nombre de fonctionnalités d'appareils électroniques qu'on utilise quotidiennement entraîne le passage d'une vision centrée sur des anciennes machines multifonctions vers des appareils variées en interaction distribués et éparpillés dans l'environnement. Sachant qu'un système est un ensemble intégré d'éléments (produits, personnels, processus) connectés et reliés entre eux, en vue de satisfaire, dans un environnement donné, un ou plusieurs objectifs définis et ayant des caractéristiques comme les composants qui le constituent, les relations entre ces composants, son environnement, les contraintes qu'il subit, les évolutions au cours du temps. La combinaison de ces derniers nous conduit à qualifier certains systèmes comme étant complexes dû à l'hétérogénéité des composants les constituant, à leurs évolution à diverses échelles de temps et à leurs répartition géographique intégrant des systèmes numériques, physiques et/ou des opérateurs humains dans la boucle. La difficulté d'avoir une bonne vision du système quand il est complexe (dispositifs réels et d'autres simulés) et la probabilité d'erreur de conception importante nous amène à réfléchir sur la possibilité de spécifier le produit et vérifier la conception à l'aide d'un prototype virtuel, on parle de simulation.

Quand un système complexe nécessite l'emploi de différents composants spécifiés par différents concepteurs travaillant sur des domaines différents, ceci augmente fortement le nombre de prototypes virtuels. Ces différents composants ont malheureusement tendance à demeurer trop indépendants les uns des autres empêchant ainsi à la fois les différents concepteurs de collaborer et leurs systèmes d'être interconnectés en vue de remplir une ou plusieurs tâches qui ne pourraient pas être accomplies par l'un de ces éléments seulement. Le besoin de communication et de coopération s'impose. Cela doit tenir compte des différents acteurs et les coordonner dans leurs interactions au sein de ce système complexe. Or les avancées en simulation dans chacun des domaines sont considérables, chacun disposant de ses propres logiciels. Des solutions d'interopérabilités sont donc nécessaires pour la mise en oeuvre d'une co-simulation encourageant le dialogue entre les disciplines et réduisant les erreurs, le coût et le temps de développement.

Dans notre thèse nous participons à la conception d'un système de co-simulation qui intègre différents outils de simulation-métiers basés sur la modélisation du comportement de dispositifs comme la simulation énergétique et la simulation d'usure de matériaux de construction au sein de la même plateforme

Après la prise en compte des notions d'architecture, de communication (entre les simulateurs ou avec les utilisateurs) et de visualisation pour définir les modèles d'architecture. Nous analysons l'architecture gérant l'interopérabilité. Nous proposons une approche d'interopérabilité se basant sur la réutilisation et l'échange de composants de calculs. Nous aborderons successivement les problématiques liées aux niveaux structurel et sémantique d'interopérabilité, aux stratégies co-simulation, aux méthodes de conception du modèle de tâches permettant la construction de composants *boite noire*. Puis nous présenterons la mise en application concrète de notre méthodologie de conception globale et de l'outil de vérification de certaines propriétés de l'architecture, comme la cohérence et la sémantique.

Mots-clés : Co-simulation Interoperabilité Design centrée utilisateur Prototypage rapide

Abstract

The large number of electronic device features we use on a daily basis means a shift from a vision of old multifunction machines to distributed, widely distributed distributed devices in the environment. Knowing that a system is an integrated set of connected and interrelated elements (products, people, processes) in order to satisfy, in a given environment, one or more defined objectives and having characteristics such as the components that constitute it , the relations between these components, its environment, the constraints it undergoes, evolutions over time. The combination of these leads us to qualify some systems as complex due to the heterogeneity of the components constituting them, their evolution at various time scales and their geographical distribution integrating digital systems, physical and / or human operators in the loop. The difficulty of having a good vision of the system when it is complex (real and other simulated devices) and the probability of significant design error leads us to reflect on the ability to specify the product and verify the design using a virtual prototype, we are talking about simulation.

When a complex system requires the use of different components specified by different designers working on different domains, this greatly increases the number of virtual prototypes.
These different components unfortunately tend to remain too independent of each other thus preventing both the different designers from collaborating and their systems from being interconnected in order to fulfill one or more tasks that could not be accomplished by one of these elements only. The need for communication and cooperation is needed. This must take into account the different actors and coordinate them in their interactions within this complex system. But the advances in simulation in each area are considerable, each with its own software. Interoperability solutions are therefore necessary for the implementation of a co-simulation encouraging dialogue between disciplines and reducing errors, cost and development time.

In our thesis we participate in the design of a co-simulation system which integrates different tools of simulation-trades based on the modeling of the behavior of devices like the simulation energetics and the simulation of wear of building materials within the same platform.

After taking into account the concepts of architecture, communication (between simulators or with users) and visualization to define architecture models. We analyze the architecture that manages interoperability. We propose an interoperability approach based on the reuse and exchange of computational components. We will successively address the issues related to the interoperability structural and semantic levels, the co-simulation strategies, the design methods of the task model allowing the construction of *black box* components. Then we will present the concrete implementation of our global design methodology and the verification tool of some properties of the architecture, such as coherence and semantics.

keywords: Co-simulation Interoperability User-centered design Rapid prototyping

List of acronyms

AI: Artificial intelligence
AIKM: AI Knowledge Map
ALSP: Aggregate Level Simulation Protocol
AMAS: Adaptive Mutli-Agent Systems
AMOEBA: Agnostic Model builder by self Adaptation
API: Programming Interface
AUTOSAR: AUTomotive Open System ARchitecture
BCVTB: Building Controls Virtual Test Bed
BIM: Building Information Modeling
BPMN: Business Process Model and Notation
CCSIL: Command Control Simulation interface Language
COM: Component Object Model
CORBA: Common Object Request Broker Architecture
CS: Co-simulation
DAE: Algebraic Differential Equations
DAI: Distribution of Artificial intelligence
DEVS: Discrete Event System Specification
DIS: Distributed interactive simulation
DLL: Dynamic Link Library
DOD: Departement Of Defense
DREAM: Dynamic Data Relation Extraction using Adaptive Multi-Agent System
FIFO: First In First Out
FMI: Functional Mockup Interface
FMU: Functional Mockup Units
FOM: Federation Object Model
GDS: Generic Data System
HCI: Human Computer Interaction
HDL: Hardware Description Language
HLA: High Level Architecture
IAI: International Alliance for Interoperability
ICAR: Interface for Component Architecture
ICT: Information and Communication Technologies
IDL: Interface Description Language
IEEE: Institute of Electrical and electronics Engineers
IFC: Industry foundation classes
IP: Internet Protocol
IT: Information Technology
JAR: Java archive
LCIM: Levels of Conceptuals Interoperability Models
MAS: Mutli-Agent Systems
MBDM: Model Based Data Management
MDA: Model Driven Architecture

MDI: Model Driven Engineering
ME: Model Exchange
MECSYCO: Multi-agent Environment for Complex System CO-simulation
MPI: Message Passing Interface
MUSE: Multiple Unified Simulation Environment
NECSI: New England Complex System Institute
NCS: Non Cooperative Situations
ODBC: Open Database Connectivity
ODE: Ordinary Differential Equations
OMT: Object Model Templates
OWL: Web Ontology Language
PDE: Partial Differential Equations
PDU: Protocol Data Units
QSS: Quantized State System
RDF: Resource Description Framework
RTI: Runtime infrastructure
SEDRIS: Synthetic environment Data Representation & Interchange Specification
SIMNET: Simulation Networking
SPARQL: SPARQL Protocol and RDF Query Language
SSGM: Subsystem to Subsystem Graph Model
TCP: Transmission Control Protocol
UDP: User Datagram Protocol
UML: Unified Modeling Language
USAFE: United States Air Forces in Europe
UX: User eXperience
XML: Extensible markup language

Contents

List of Figures

CHAPTER I: Introduction and motivations

1 work environment

The "smart city" is no longer a futuristic dream, the impact of digital in urban development is now a reality. In 2016, it is estimated that 54,5% of the worlds population is urban and the urban population is expected to continue to grow so that by 2050, it is expected that 7/10 of the population will live in urban centers [Group, 2014]. Due to this increase of urban population, the challenge of rethinking the concept of a city sprang. As Laurence Lafont, Public Sector Director at Microsoft France, said: "Digital is not an end in itself; the question is rather how will it help cities to transform and develop". The smart city proposes to use Information and Communication Technologies (ICT) to design technologies which should respond to people's needs through sustainable solutions for social and economic aspects [Albino et al., 2015]. As computer designers, our attempt to make our university campus smart and sustainable as part of the neOCampus initiative is fraught with problems. The desire to study particular aspects of smart cities illustrates the need for interdisciplinary work.

The neOCampus operation, which is our first and main motivation in this work, aims to study and implement the concepts of Smart Cities in the University campus of Toulouse III Paul Sabatier. Many initiatives have been launched, mainly the study of the impact of human activity on the campus footprint. But these applications have to face technological challenges such as their heterogeneity, their spatial distribution or the number of entities involved. These characteristics make it mandatory to consider smart cities as complex systems and to approach them with an interdisciplinary approach.

work complexity Designing an interactive system is a difficult task. Designing and evaluating a so-called "complex" system is even more so.

From a software point of view, a system can be seen as an integrated set of elements interconnected with each other, in order to satisfy in a given environment one or more pre-defined objectives. The components of this system include both the facilities, the hardware and software equipment, the data, the services, the qualified personnel and the techniques necessary to achieve, provide and maintain its efficiency.

A complex system has many characteristics such as the heterogeneity of its components, their evolution at different time scales and their geographical distribution integrating both digital systems, physical and/or human operators. These complex systems are usually decomposed into subsystems, following either:

- top-down approach, if we have a holistically vision of a system (understanding each part of the system by first understanding the whole system)

- bottom-up approach - from existing components that need to be reused - if we have a reductionist vision of a system (introduced by R. Descartes, where we try to understand a

"system from the study of its individual components).

Three characteristics that determine a complex system can be distinguished, according to [Guckenheimer and Ottino, 2008]:

- Its complex structure into several components and subsystems. We may also have a network that describes which components of a system interact.

- The behavior possibly emerging and interactions "not obvious" from the analysis issues of the subsystem. We frequently use the term emergent to describe behaviors that arise from the interaction of subsystems and are not evident from analysis of each subsystem

- Its adaptation as well as its evolution over time are characteristic of critical infrastructure systems and fundamental to the life sciences.

Complex systems are everywhere, starting by the universe, Earth's global climate, organisms, the human brain, social and economic organizations and traffics. We usually focus on their characteristics while a unanimously accepted definition of their nature remains to be formalized. We also use the term **system of systems** alternatively when the functionality provided results from the composition of pre-existent systems with non-trivial interconnections. It offers more functionality and performance than simply the sum of the constituent systems. [Maier, 1998] distinguishes five traits for identifying system of systems challenges:

- Operational Independence of Elements: The component systems of the system of systems must be able to usefully operate independently. In other words, the components themselves fulfill the client-operator functions.

- Managerial Independence of Elements: Component systems are acquired and integrated separately, but maintain a continuous operational existence, independent of the system of systems.

- Evolutionary Development: A system of systems is never completely formed or complete. The development of these systems evolves over time and their structure, function and purpose are added, removed and modified as experience with the system develops and evolves over time.

- Emergent Behavior: The functions performed and purposes carried out don't reside in any component system. They're emergent properties of the entire system of systems. The principal purposes supporting engineering of these systems are fulfilled by these emergent behaviors.

- Geographical Distribution of Elements: Systems can easily exchange only information and knowledge with one another, and not substantial quantities of physical mass or energy

As a result, we are led to the inability of these systems (subsystems) to speak the same language to each other and to share information effectively and correctly - **to be interoperable** - which is one of the major problems that complex systems frequently face. To achieve this cooperation on a global level, it seems important to opt for an open environment that allows continuous dialogue between the different parties.

Before going further, one should differentiate between *interoperation* dealing with **how** a complex system works, how its components exchange with each other and how they are orchestrated to hand over the required functionality to the user. The composition, on the other hand, focuses on **what** components and functionality can be integrated into systems without causing any failure, or creating problems with other components.
Following the categorization of [Page et al., 2004], admitting that *composability* deals with models, *interoperability* deals with their implementation, and *integrability* deals with the hardware and configuration side of connectivity.

As the integrability deals with the physical and technical connections between systems. This communication protocols exist for exchanging data between participating systems, and the communication infrastructure is established allowing systems to exchange bits and bytes, and that the underlying networks and protocols are unambiguously defined. We estimated that our contribution will not be on this low level. On the other hand, as we were dealing with models that some of them pre-exist and others are developed separately by experts in a parallel time, that conceptual models are unfortunetly not all documented based on engineering methods enabling their interpretation and evaluation by other engineers. We didn't consider the alignment of issues on the modeling level (composability).
Thus, we focused our work on the interoperability dealing with the implementation details of interoperation, which includes the exchange of data elements via required and provided interfaces, the use of middleware, mapping to common information exchange models.

We believe that a complex system (and its behavior, since the temporal dimension is indeed present) includes more than the sum of its individual components, possessing strong synergies between the sub-parts. In attempting to interoperate heterogeneous systems, a systemic approach based on holism must be taken into account.

Thus, to understand the functioning of the system as a whole, we must simultaneously study all its components and their interactions and not study each component in isolation. We distinguish the case where the components already exist, made by experts, so we focus on the interactions only. The other case is when we don't have the component and we need to develop it, so the component's component is a matter of fact. In addition, this holistic development process [Van der Auweraer et al., 2013] where the partial solutions developed independently are integrated sooner in the process and more frequently is the only way to achieve an innovative and optimal multi-disciplinary solutions.

2 Modeling and Simulation

The modeling and simulation (**M&S**) is a substitute for physical experimentation, in which computers are used to compute the results of some physical and not physical phenomenon.As it's apparent from its name "Modeling and simulation":

- **Modeling** is the process of producing a model; the model is, according to the US-DOD, all physical, mathematical, or other logic representation (abstraction) of a system, entity, process or a physical phenomenon. A model is similar to but can be simpler than the system it represents.
  When a model is intended for a simulation study, we speak of a *mathematical model* developed with the help of simulation software. There are different ways to make a model classification. This could be done by system type, or by type of mathematical construction. This last way is more interesting since it deals with the mathematical and statistical aspects of the models, rather than how to represent a particular type of system. Acoording to modelia[1], we distinguish a partial list of a mathematical model types that are pretty widely used:

  - Deterministic (input and output variables are fixed values)
  - Stochastic (at least one of the input or output variables is probabilistic)
  - Static (time is not taken into account)
  - Dynamic (time-varying interactions among variables are taken into account)
  - Based on Partial Differential Equations (PDE) (systems vary according to several continuous variables)
  - Individual-based (A set of individuals, rules of behavior, functions of the local environment) similar types are:
    - Cellular automata where the individuals are placed on the intersections of a network, and each intersection is occupied
    - Multi-agent model where agents are the simulated individuals

- **Simulation** is defined according to IEEE Standard Glossary of Modeling and Simulation as an implementation of one or more models evolving over time and made for a specific purpose. It's a tool to evaluate the performance of a system, existing or proposed, under different configurations of interest and over long periods of real time.
  Simulation aims to reduce the chances of failure to meet specifications, to prevent under or

[1] `http://www.modelia.org/html/050929_formationIntroModDec2005/supportsDesInterventions/WallachDcomment_TypesOfModel_deQuoiSagitIl.pdf`

over-utilization of resources, to eliminate unforeseen bottlenecks, and to optimize system performance.

When studying the dynamics of systems to understand the behavior of complex systems over time, as in our case, two techniques are distinguished:

- **Discrete event simulation** is appropriate for systems having a continuous time base but inputs and state transitions occur at discrete time instances. The state transition consists of two parts: The external transition function which is executed whenever an input arrives and the internal transition function where the times of execution are scheduled by the time advance function. An example of such a system is the number of students in a library: The number of students is discrete (integer) and the number of students only changes when someone enters or leaves the library.

- **Continuous simulation** is appropriate for systems with a continuous state that changes continuously over time. An example of such a systems is the amount of liquid in a tank and or its temperature. So if we observe the water that heats, we do not consider any event: the water heats permanently, continually, and we choose to observe its temperature every second, every minute, every millisecond. No matter when one observes it, the water will have a temperature - according to which a decision can be made. Such a system can be described by differential equations. Continuous simulation is a technique to solve these equations numerically.

Models can be expressed in a variety of formalisms that can be understood as means for specifying subclasses of dynamic systems. These formalisms are chosen according to the objective of the study to be realized. Formal models, based on concepts and mathematical properties, have two dimensions: space and time [Zacharewicz, 2006].

We thus distinguish two classifications: **the temporal classification** whose representation of time can be discrete or continuous and **the spatial classification** where the space whose descriptive variables can take values in a finite or infinite set.

Keep in mind that digital computer simulation involves the discretization of time, space or both. Certain formalisms already have a discrete component, in the opposite case, it will be necessary to realize the discretization of the time or the space. For models described by differential equations[2], time is discretized in digital integration techniques.

The CHEAr[3] center distinguishes the most common simulation techniques based on the fields of application and composition criteria:

- Numerical simulation: purely software, not real time

[2]Association of a time base and variables also continuous, the equations define the changes of states
[3]Centre des hautes études de l'Armement

- Hybrid simulation: it makes the digital and the continuous interact
- Interactive simulation: involves the human factor, war games
- Virtual simulation: piloted simulation, Human in the loop, reproduction of the real commands
- Instrumented simulation: integrates real hardware
- Real time simulation: when the respect of the temporal constraints in the execution of the treatments is as important as the result of these treatments

Note that some of these techniques overlap; Hybrid simulation, virtual simulation and instrumented simulation are real-time simulations. Numerical simulation can be interactive. Piloted simulation and instrumented simulation in many cases are interactive.
With these different types of simulations, their interoperability has emerged as a fundamental technique to enhance their applicability and minimize the cost of development and maintenance.

Simulation can show virtually (creating virtual scenes) how a real physical situation would happen.
According to [Jara et al., 2010] computer modeling requires an analysis of the problem with the identification of the variables and the algorithms, and its implementation on a specific software platform. A computer simulation is an implementation of a model that allows us to test the model with different initial conditions with the objective of learning about the model's behaviour. The applicability of the results of the simulation to those of the real system depends on how well the model describes reality. see(cf. Figure 1).

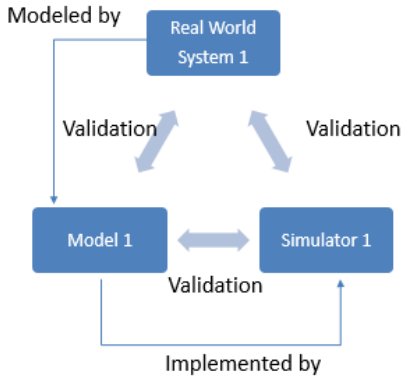

Figure 1: modeling and Simulation representation

Since the modeling and simulation (**M&S**) [Friedman and Ghidella, 2006] doesn't promote a holistic development process, they may only improve the development of the partial solutions, because of the difficulty to integrate models developed by different specialized tools or those with Intellectual Property.
Co-modeling could be an alternative way, assuming that models are described in a unified language, and then simulated. But each domain has its own particularities when it comes to simulation, making it impractical to find a language and simulation algorithm that fits all.

Co-simulation (cf. Figure 2) is defined, to overcome these challenges, as the coupling of several simulation tools where each tool deals with a part of a modular problem, where the data exchange is restricted to discrete communication points and where the subsystems are solved independently between communication points. This allows each designer to interact (and work on a subsystem) with the complex system in order to maintain its business expertise and continue to use its own digital tools. This designer can therefore use a variety of specification tools at different levels of abstraction. In addition, the design of the models may be subject to a competitive design, as teams design in different design times. Some may have finished while others pursue it, so competition is stimulated and motivates teams to speed up their activities.
A co-simulation, according to [Palensky et al., 2017], is composed of a set of coupled simulators that cooperate with each other. Each simulator has its own solver and works simultaneously and independently on its own model. The simulators are coupled by dynamically connecting the models using their input and output variables, so that the outputs of one simulator become the inputs of the other and vice versa. The variable exchange, time synchronization and execution coordination is, in the most general case, facilitated in runtime by a so-called master algorithm, which orchestrates the entire co-simulation. Aside from the validation aspects, software interoperability is often not given or possible, numerical phenomena and problems are sometimes even unsolvable.
As in our case where we're dealing with complex, heterogeneous systems that can neither be investigated in analytical nor in purely experimental fashion. Data exchanged and utilized between simulations has come to define their interoperability, such as messages ex-changed between simulators or actions defined between units on the strategic maps. Thus, interoperability of the various simulators requires standardized interfaces.

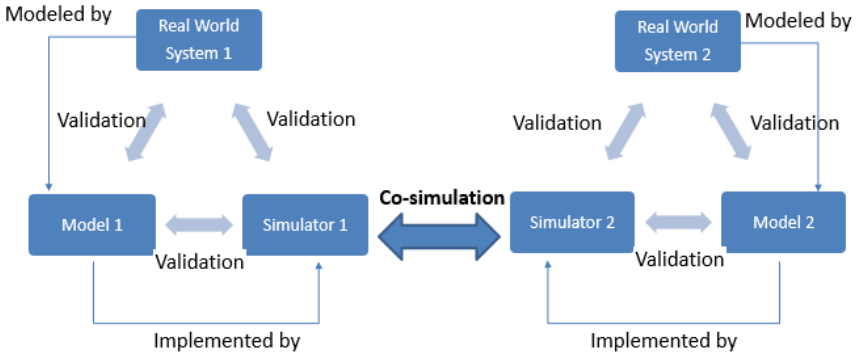

Figure 2: Overview of co-simulation of two subsystems

A complex system is also characterized by many people involved in its life cycle that includes design, development, manufacturing, operations, maintenance and decommissioning. That comes down to give extensive attention to the user characteristics, usability goals, environment, etc.

Among the constraints of the complexity to which we confront ourselves in our work, the management of the problems of organization in these units, with a significant human component, participating in the neOCampus operation. The human operator brought to coexist and interact with our complex system, is at the same time, by its adaptive capacities the guarantor of the reliability, the flexibility and the reactivity, and paradoxically one of the main causes of insecurity. Uncertainties related to his behavior and his lack of knowledge of the global system hinder his good interaction with it and may reduce performances.

Long regarded as secondary, the human operator however takes more and more importance and proves, at the same time, the most difficult to control. Moreover, a part of our work addresses human-machine interaction, helping to understand and cooperate heterogeneous research units by focusing on the interactions of the human operator with the system in order to make the best use of its capabilities. Some of the principles that will ensure a design is human centered cited by [Endsley, 2016] are:

- The design is based upon an explicit understanding of users, tasks and environments. This comes down to know who are the users and what are the users tasks and goals.
- Users are involved throughout design and development.
- The design is driven and refined by user-centered evaluation.
- The process is iterative. A process for arriving at a decision or a desired result by repeating rounds of analysis or a cycle of operations.
- The design addresses the whole user experience. By understanding the user's experience levels.
- The design team includes multidisciplinary skills and perspectives. Knowing for example if the user is multitasking.

In our thesis we participate in the design of a co-simulation system which integrates different tools of simulation-trades based on the modeling of the behavior of devices like the simulation energetics and the simulation of wear of building materials within the same platform.

After taking into account the concepts of architecture, communication (between simulators or with users) and visualization to define architecture models. We analyze the architecture that manages interoperability. We propose an interoperability approach based on the reuse and exchange of computational components. We will successively address issues related to the structural and semantic interoperability levels. Using, in an original way, respectively interoperability standards and a dynamic data mediation based on artificial intelligence, co-simulation strategies, task model design methods allowing the construction of *black box* components, based on an original user centered design approach. Then we will present the concrete implementation of our global design methodology and the verification tool of some properties of the architecture, such as coherence and semantics.

In Chapter II, we present the issues related to the smart campus design process and will focus on the issues and concept of interoperability between campus business tools for dynamic simulation. We will introduce the existing interoperability solutions (data, models and computation codes) and we propose a new interoperability approach that will be used at each stage of the design process. and co-simulation tools

In Chapter III, we will be interested in the co-simulation of heterogeneous tools. We will define the co-simulation, and study the synchronization models (How data will be exchanged between models), and compare the advantages and the limits allowing us to make a specific choice to our context. Following which we will define the steps to build our framework.

In Chapter IV, we will be interested in the co-simulation framework interoperability and will detailed the one based mediation model and described the other based dynamic data mediation. For this last one we will present the adaptive multi-agent systems, then we will compare the two approaches.Finally, we will present the practical application of our global design methodologies on a multi-stakeholder, multi-business project. This is the *neOCampus* operation where we worked in close contact with many experts, as well as other industrial and academic partners.

In Chapter V, we will be interested in the conception methods for and with the user, then we will explain the needs to achieve a user centred design approach. This part of the work was a consequence of the interaction of neOCampus users with our framework. As they didn't know how to generate the components needed to be plugged in our plateform from their own simulations. We will focus on the tasks model designed and interface developed for the black-box components generation. Finally we will present the interface advantages deducted by the statistical analysis made for that purpose following a HCI "Human Computer Interaction" approach.

Chapter VI will be the last chapter, contributions will be analyzed and a conclusion will be given as well as future work.

CHAPTER II: Interoperability and Issues in the smart campus process

1 Building a Smart campus at Toulouse III Paul Sabatier University

1.1 Smart campus Concept

The societal, energy and economic context drives lot of Universities to place at the heart of their project of establishment another project: that of a smart campus, in partnership with all the political and economic actors of the territory, without forgetting the universities community (students, staff, teachers and researchers). This smart campus project has as missions to improve the university's environmental impact to make it a sustainable and responsible campus, to evolve the campus towards a smart, digital, connected and responsible campus and to put it as an integrated element in a larger whole: a smart campus in a smart city. As a matter of fact, the Smart Cities concept is more than an economical approach to reduce maintenance costs of cities: it is a socio-ecological revolution, which intends to improve the symbioses between a city, the environment and the citizens [Ahvenniemi et al., 2017].

This concept of smart cities, or at small scale smart campus is original and systemic in its:

- Design: Smart campus crosses human constraints and opportunities with scientific and technological capabilities.
- Thinking: All research laboratories, in line with their areas of expertise, are working on experimenting with the smart campus of tomorrow.
- Implementation: An impressive number of actors from all strata of the university (students, researchers, teachers, administrative staff ...) participate.
- Scope: The entire territory collaborates and promotes a territorialized and decompartmentalized approach to the sustainable city.

The smart campus concept has in general point of view, according to [Verstaevel et al., 2017], four fundamental aspects:

- The use of **information and communication technologies** (ICT) to manage the city's assets, improving the efficiency of the services offered by the city.

- The improvement of the **quality of life** of residents by a digital transformation of their working and living environments.

- The central roles of **citizens** in this transformation, by orienting cities towards citizens needs and by an active implication of citizens in cities decisions thanks to the usage of ICT.

- The **inter-disciplinarity** aspect of researches on Smart Cities, implying many domains such as urbanization, social science or informatics.

1.2 Smart campus: complex system

We mentionned in the part 1 the focus on the data exchange. Data are the core of the smart campus, everything relies at some point on the data that are collected in the campus. As described in [Verstaevel et al., 2017] the ongoing projects of neOCampus, highlighting the cooperation between scientists from different fields and the mutual benefits for researchers management, and technical staffs, face some technological challenges such has their heterogeneity, their spatial distribution or the number of entities involved. Those characteristics make mandatory to see Smart Cities as complex systems.
Also from an Information technology (IT) designer point of view, smart campus applications share common properties which makes them complex systems [Harrison and Donnelly, 2011] [Verstaevel et al., 2017]:

- **Large-scale**: Due to the amount of entities (physical and virtual) involved in the Smart Campus, Smart Campus IT applications reach unprecedented scale in many dimensions such has the number of lines of code to develop an application, the amount of data stored, accessed, manipulated, and refined, the number of connections and interdependencies or the number of people involved and interactions that may occurred. We may even talk of Ultra-Large-Scale Systems.

- **Non-Linearity**: In Smart Campus, even the smallest causes can have large effects. For example, a change in the timing of a traffic light may results in huge traffic congestions. A non-linear system is a system in which the change of the output is not proportional to the change of the input. This non-linearity is a huge problem for ICT as it may lead to unpredictable or counter-intuitive situations, which makes the task of controlling these systems very complex.

- **Heterogeneity**: Smart Campus are composed of various heterogeneous devices. This heterogeneity implies challenges at various levels. The observation of Smart Campus through

a large scope of sensors, each sensor independently designed to observe a specific feature, results in producing large volumes of heterogeneous data. Those Big Data require new infrastructures to enable the exchange and storage of those information, but also requires to create new tools and norms to manage them. Indeed, sampling rates, data scales or data formats are as various as sensors are numerous. But this heterogeneity might also be a source towards innovative solutions.

- **Openness**: The openness property characterizes the ability of a system to deal with the appearance and disappearance of some of its parts. As Smart Campus are in constant urban mutations, ICT applications must deal with the appearance and disappearance of devices, and the data and action associated with. Openness is a crucial challenge for the large acceptance of new technologies and a key towards sustainability.

- **Unpredictable Dynamics**: Users activity is a huge source of those unpredictable behaviours in Smart Campus. This unpredictability involves to provide to IT systems the ability to continuously self-adapt to changes that may occurs in the dynamics of the city.

- **Spatial Distribution**: At the opposite of centralized IT systems, where information is sent to a central node which takes the decisions and exercises control over the different components, , the spatial distribution of entities in Smart Campus invites to rest on the autonomy of the entities, decentralizing the control over the different entities.

- **Privacy**: Smart Campus are able to collect and gather large amount of information, and this could harm the privacy of end-users [Martínez-Ballesté et al., 2013]. Privacy rises questions at various level, from the designer of IT applications to its users, transcending the previously unidirectional relationship between designers and users. The control by citizens in the behaviour of Smart Cities is probably a key to ensure this privacy, but allowing such control involves an ethical design of IT application. This implies new development methodologies taking from the very beginning of the design of an application the privacy into account.

This list of properties is not exhaustive, but it illustrates the complex nature of designing IT application in Smart Campus. Addressing those challenges and properties is more than an IT problem, it requires a multidisciplinary approach, and involves many scientists from different fields.
Many Smart Campus initiatives may be found in the scientific literature, each of them focusing on particular aspects. We can cite Lancaster University, where the focus is made on socio-digital sustainability [Bates and Friday, 2017]. At Lisbon, for example, the focus is made on energy efficiency [Gomes et al., 2017]. A prototype of mobile social networking system is deployed on campus in [Yu et al., 2011], and several applications are implemented based on the proposed architecture to demonstrate the effectiveness of the architecture. [Rohs and Bohn, 2003] focuses on the aspect of linking virtual and physical elements in such a setting and present the ETHOC system, which enables users to attach virtual counterparts to printed material. This system allows to put ubiquitous computing concepts into practical use and to gain new insights into the design of virtual counterparts of

real-world objects. [HUANG et al., 2012] proposes the five key technologies to support the construction of smart campuses: learning scenario recognition and environment awareness,campus mobile internet technology,social networking technology,learning analytic technology,digital resources organization and sharing technology. [Tan and Wang, 2010] designs a specific the Internet of Things application model which can apply to automatic facilities management in the smart campus. The introduction of the application of the internet of things and the cloud computing in education was made in [Nie, 2013], they then discuss the current status of smart campus and indicate the difference between digital campus and smart campus. [Atif et al., 2015] aims at developing a novel ubiquitous learning model within a pervasive smart campus environment, in order to identify the steps towards building such an environment and the involved learning processes. At University of Brescia, [De Angelis et al., 2015] used the classroom building as demonstrator of the energy strategies to improve the performance of existing buildings toward zero energy goals. Innovative systems to reduce energy consumption (i.e. heating, cooling and ventilation, lighting and electric equipment) and Smart Automation are crucial in the Smart Campus School Project of which this research is a preliminary step of development.
On the next section, we present our own initiative, entitled neOCampus.

1.3 Our Smart campus

The Toulouse III Paul Sabatier University is composed of more than 60 research structures and around 10 doctoral schools. More than 30 thousand students were enrolled in university studies in 2018. More than 6 thousand teacher-researchers, engineers, technicians and staff members in the different laboratories. The University has almost 400k m2 of built-up areas. The cost of functioning of the university represent 21% of its budget of more than 400M euros. All the activities on the campus consume 140 GWh a year and produce 23 250 tons of CO2 (diagnosis made in 2010).

The neOCampus operation [Gleizes et al., 2017], supported by the University of Toulouse III, aims to link the skills of researchers from different fields of the University to design the campus of the future. Three major areas are identified: facilitating the life of the campus user, reducing the ecological impact and controlling energy consumption. The campus is considered as a smart city where several thousand data streams come from heterogeneous sensors placed inside and outside the buildings (cf. Figure 3) (CO2, wind, humidity, luminosity, human presence, energy and fluid consumption , ...). We distinguish:

- Raw data: These are the energy consumption data (water, electricity, gas).
- Activity-specific data: These are post-processed data resulting from the merging of raw data (pedagogical activities, room occupancy, ..).
- Incident-specific data: These are the failures identified on campus (heating of computer equipment, network failures, ...)

- The ambient data: This concerns the context in which the scenario takes place (temperature, weather, CO2 level in the air).

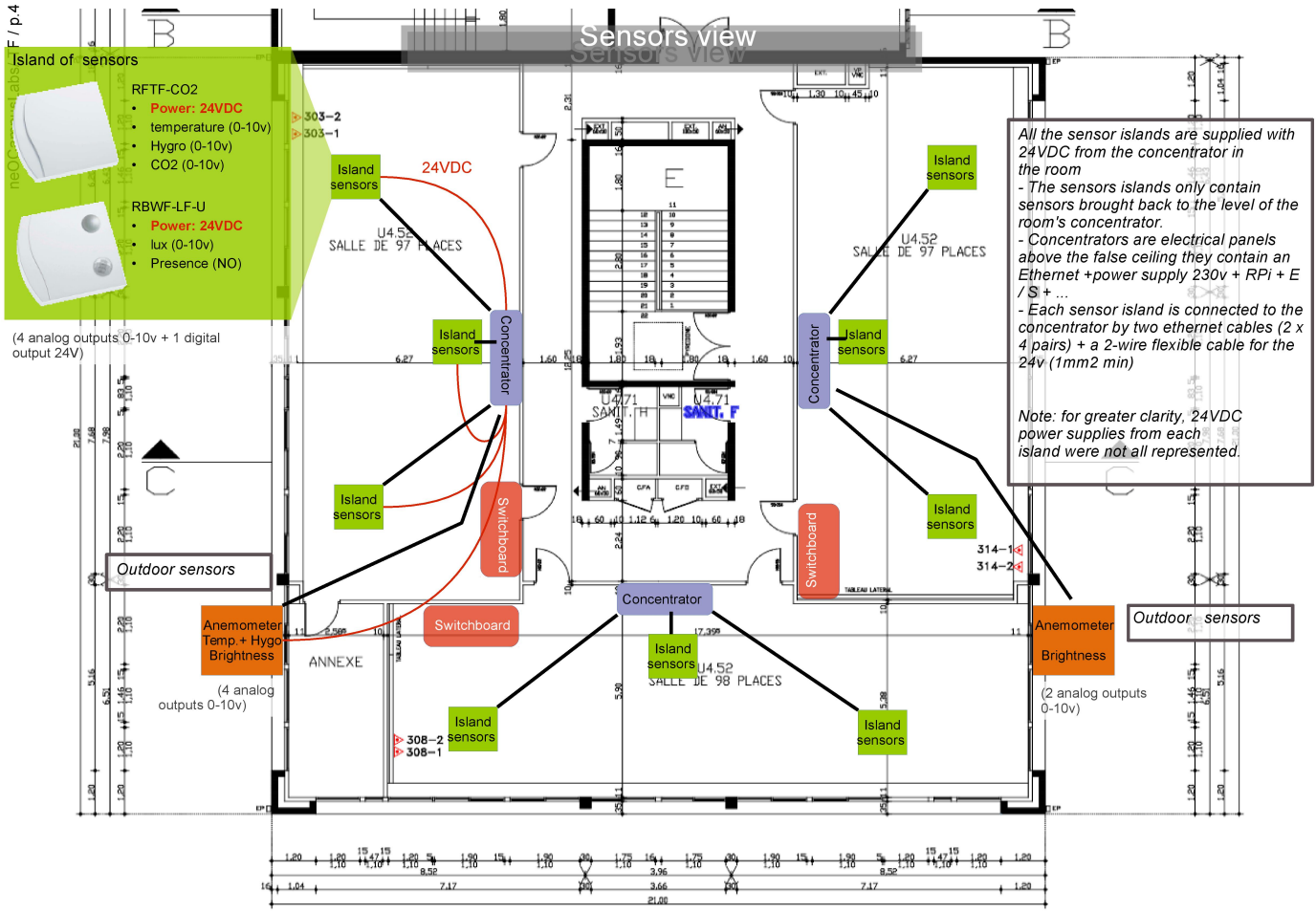

Figure 3: Overview of neOCampus sensors

These data mainly depend on the tasks performed on the system. They have different types:

- Quantifiable: data representing a numerical value (quantity, volume, ...)
- Textual: data giving information, or describing things
- Functional: data reporting on an action carried out or to be carried out

Users are in the center of the campus as they are interacting with these data, we distinguish different types of actors:

- casual actors as the outside staff, guest researchers, etc.
- Regular actors as the students and university staff.
- Frequent actors involved in campus maintenance and upkeep.

The presence of actors and their interactions with different types of data, lead us to introduce the concept of service design. In order to exploit, the more, the data and not be limited to the development of products without being used by the users, 3 types of provided services were distinguished:

- The perception support services allowing the users to visualize raw data
- The comprehension support services allowing for example a diagnosis of an energy overconsumption
- The production support services allowing for example the generation of a review based on the analysis of the user consumption

The campus as represented in (cf. Figure 4) is based on numerous interconnected hardware and software equipment combining innovative educational materials, sensors, storage and location systems, simulation systems and new materials in university buildings scattered throughout the campus (`https://www.irit.fr/neocampus/`). The campus is thus considered as a complex interactive system composed of multiple dynamic subsystems making unpredictable all the consequences of a change, even minimal.

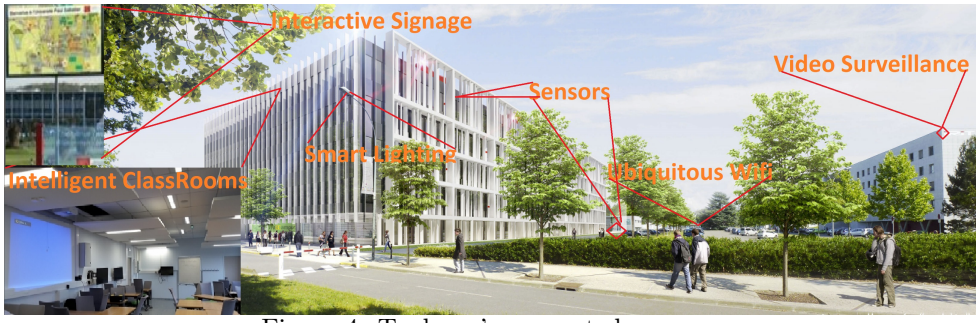

Figure 4: Toulouse's connected campus

Some ongoing projects of neOCampus were presented in [Verstaevel et al., 2017], highlighting the cooperation between scientists from different fields and the mutual benefits for researchers management, and technical staffs. Some of these projects help involving users into the ecological philosophy of Smart Campus. Scientists work on different systems depending on their expertise but when it comes to cooperate and communicate with each other they face the interdisciplinary approach's issues.

Unified global modeling can be a solution but it consists of combining all the studied systems in one and the same model. This ensures strong coupling between campus subsystems and ensures physical consistency, which allows the designer to optimize a single model containing a variety of objectives covering all of the project's systems. This is for example what is done in the Modelica BuildSysPro library [Schumann et al., 2016]

However, this type of modeling requires the presence of all the models in a single language to be able to ensure a strong coupling. If not, developers of modeling tools must redevelop the different models in a single language while ensuring physical consistency between them. This requires a great mastery and expertise in each profession. The heterogeneity of sub-models in tools and languages, and the need to access the implementation code of each model to create a unified global model is often a blocking point. The confidentiality associated with certain models leads us to resort to "black box" models (the source code is not accessible).

The neOCampus design approach is necessarily scalable and adaptive, which directs the work towards the development of global and open simulation environments dynamically linking simulators (under simulation environments running on different platforms) and heterogeneous real systems . In addition, the different parts of the system can be co-simulated at different levels of abstraction. Interoperability is therefore an important aspect to consider in the design.

2 Support the design process through interoperability

In the neOCampus operation we are often brought to work with scientists from different fields, they are modeling the systems studied and the implementation codes of these models are not accessible. To benefit from the expertise of each tool, we propose to make the models interoperable

with each other (different languages, different tools, different hypotheses) to ensure a coupling of models and data.

2.1 Definition, issues and concepts of interoperability

2.1.1 Definition of interoperability

There is lot of definitions of interoperability in literature, some define it as the ability of a computer system to work with other existing or future computer products or systems, without restriction of access or implementation [Chapurlat and Daclin, 2012]. In [Medvidovic and Taylor, 2000] interoperability is defined as a **translator** that allows systems to communicate even when the data is not the same. Interoperability can also be defined as the ability of two or more entities to communicate and cooperate despite differences in the implementation language, the execution environment, or the model abstraction [Kohar et al., 1996]. Programming Interfaces (APIs) are the basis of IT interoperability. For example, the J2EE specification for the Java programming language has many types of APIs. These APIs can be applied to different types of IT resources (databases) or applications (ERP). We may talk about interoperability only when the interfaces are completely defined, known and freely usable. The interfaces are much less complex than the systems that use them. Moreover, they are stable over long periods of time because they are independent of the evolutions of these systems. For instance, the interfaces of CORBA [Ben-Natan, 1995] distributed objects are described using an Interface Description Language (IDL). It is a software architecture defined for interoperability, assuming that the object technology is the most promising to achieve these objective. Thus CORBA interfaces consist of the object public attributes and methods [Suzuki and Yamamoto, 1999]. This object vision and according to [Panetto and Molina, 2008] defines interoperability as the ability of a software tool to act on objects whose types may be different from the arguments of the tool. This object-oriented approach of the 1980s and its "everything is object" principle, software engineering is now moving towards model-driven engineering (MDI) and the "everything is model" principle. This approach can be considered both in continuity and in break with previous work [Bézivin and Briot, 2004]. In continuity because it is the object technology that triggered the evolution towards the models. Indeed, once the conception of computer systems in the form of objects communicating with each other, the question was posed of classifying them according to their different origins (business objects, techniques, etc.). In break since it aims to provide a large number of models to express separately each of the concerns of users, designers, architects, in a more radical way than could be the approaches of patterns [Gamma, 1995] and aspects [Kiczales et al., 1997].

2.1.2 Languages to represent interoperability

Throughout the last few years a variety of techniques have been implemented to enable the interoperability of information systems. The Unified Modeling Language (UML) [D'souza and Wills, 1998], the Business Process Model and Notation (BPMN) [White, 2008], the Extensible Markup Language (XML) [Bray et al., 1997] and the Language Definition Interfaces (IDL) [Lewis et al., 1998], are all mechanisms that allow for syntactic interoperability. In addition, the REST and SOAP web services [Mumbaikar et al., 2013], and more generally the service-oriented architecture,

allow interoperability at the technical level. We also note the Common Object Request Broker Architecture (CORBA) [Siegel and Frantz, 2000], the Component Object Model (COM) [Williams and Kindel, 1994] or the Open Database Connectivity (ODBC) [Geiger, 1995] which are as much middleware allowing technical interoperability[4] between the distributed objects. Model Driven Architecture (MDA) [Kleppe et al., 2003] is also a tool for technical interoperability.

2.2 Model-Based Interoperability

Some works have been done in the last decades to increase the interoperability of Entreprise Information Systems (EIS) using standardized modelling approaches. [Zacharewicz et al., 2017] defined a Model-based interoperability approach which relies on the use of a common language provided by the relatively unambiguous expressiveness of formal or semiformal models as the basis for making EIS's interoperable. This last distinguishes different abstraction levels: Business, Process, Service and Data and Enterprise which require different models that involve different categories of information. To reduce this gap, model-based reconciliation can be considered as an issue. Several model driven methods have emerged to support the model transformation and at the end to facilitate the trans-level interoperability. Model Driven Architecture (MDA) is the most used one. MDA is a system design approach developed by the Object Management Group (OMG) for the development of software systems. It is related to multiple standards, including the Unified Modeling Language (UML), the Meta-Object Facility (MOF), XML Metadata Interchange (XMI), Enterprise Distributed Object Computing (EDOC), the Software Process Engineering Metamodel (SPEM), and the Common Warehouse Metamodel (CWM). It provides a set of guidelines for the structuring the specifications, which are expressed as models.
Ducq and al. 2007
defined Model Driven Interoperability (MDI) Method as a model-driven method that can be used for two enterprises that need to interoperate not only at the code level but also at Enterprise Modelling level with an ontological support with the final aim of improving their performances. This was done from the conceptual, technological, and especially from methodological point of view Refinement of the developed MDI Framework to support interoperability following a MDA approach in order to be focused on Interoperability Model. It uses model transformations to achieve interoperability defining models and an Interoperability Model at different levels of abstraction according to an MDA approach and dividing the Common Information Model (CIM) level into two sub-levels, that is to say, Top CIM level (TCIM) and Bottom CIM level (BCIM).

Due to the nature of our problem described in 1.2, we are not developing everything from scratch, some of the components we may use already exist. The ability of our system to deal with the appearance and disappearance of some of its parts (Openness). The large scope of sensors independently designed to observe, for each-one, a specific feature, results in producing large volumes of heterogeneous data. The different kind of users and applications relying at some point on the data that are collected in the campus.
We define, in our turn, interoperability as the ability to communicate heterogeneous and distributed tools or applications by adapting their data to make them cooperate in the fulfillment of tasks.

[4]technical interoperability also called structural interoperability by others, it means that simulation systems can talk to each other and exchange data but to understand each other correctly and co-simulate effectively requires substantive interoperability also called semantic one

Interoperability is a complex problem. There are many ways to deal with it. We have identified two main approaches:

- based on levels of conceptual interoperability models (LCIM)
- based on structural and semantic interoperabilities

2.3 levels of conceptual interoperability models (LCIM)

In [Diallo et al., 2011], LCIM is used as the theoretical backbone for developing and implementing an interoperability framework that supports the exchange of XML-based languages. They defined 7 levels of LCIM, as described in (cf. Figure 5), with the goal to separate model, simulation, and simulator in order to better understand how to make models interoperate. For all the levels mentioned there are corresponding reference engineering approaches, and the alignment of the technical structures with conceptual ideas will enable to cover the conceptual level of interoperability.

These levels of interoperability are ranging from no interoperability to full interoperability. In the technical domain, various models for levels of interoperability already exist and are used successfully to determine the degree of interoperability between information technology systems. However, such models are not yet established in the domain of conceptual modeling. LCIM deals with various levels of conceptual interoperability that goes beyond the technical reference models for interoperable solutions. It is intended to become a bridge between the conceptual design and the technical design for implementation, integration, or federation. It should also, according to [Tolk and Muguira, 2003], contribute to the standardization of Vérification & Validation (V & V) procedures as well as to the documentation of systems that are designed to be federated. LCIM helps to determine in the early stages of the federation development process whether meaningful interoperability between systems is possible.

Two broad functions are served by LCIM:

- The describing role of what level on interoperability exists within a composition of systems
- The prescribing role of the methods and requirements that must be satisfied during the engineering of a complex system in order to achieve a desired level of interoperability

The authors of [Rezaei et al., 2013] presented an overview of the development of interoperability assessment models. They proposed an approach to measure the interoperability and used four interoperability levels to define a metric.

Levels	Prescription of Requirements to reach this Level	Common Reference Engineering Approaches
L6(Conceptual)	A shared understanding of the conceptual model of a system (exposing its information, processes, states, and operations).	DoDAF; Military Mission to Means Framework; Platform Independent Models of the Model Driven Architecture; SysML
L5(Dynamic)	The means of producing and consuming the definitions of meaning and context is required.	Ontology for Services; UML artifacts; DEVS; complete UML; BOM
L4(Pragmatic)	A method for sharing meaning of terms and methods for anticipating context are required.	Taxonomies; Ontology; UML artifacts, in particular sequence diagrams; DEVS; OWL; MDA
L3(Semantic)	Agreement between all systems on a set of terms that grammatically satisfies the syntactic level solution requirements is required.	Common Reference Model, such as C2IEDM and CADM; Dictionaries; Glossaries; Protocol Data Units; RPR FOM
L2(Syntactic)	An agreed-to protocol that all can be supported by the technical level solution is required.	XML; HLA OMT; Interface Description Language; CORBA; SOAP
L1(Technical)	Ability to produce and consume data in exchange with systems external to self is required.	Network connection standards such as HTTP; TCP/IP; UDP/IP etc.
L0(No)	NA	NA

Figure 5: Levels of Conceptual Interoperability Model

2.4 Structural and semantic interoperabilities

[Chen, 2006] defined a framework to identify the basic dimensions regarding the enterprise interoperability and to define its research domain as well as to identify and structure the knowledge of the domain (cf. Figure 8).
For this, three basic dimensions were identified:

- **Interoperability concerns** where the content of interoperation that may take place at various levels of the enterprise is defined. In the domain of Enterprise Interoperability, the following four interoperability concerns are identified: data, service, process, and business [Guglielmina and Berre, 2005].

- **Interoperability barriers** which is a fundamental concept in defining the interoperability domain. Many interoperability issues are specific to particular application domains. These can be things like support for particular attributes, or particular access control regimes. Nevertheless, general barriers and problems of interoperability are already addressed in [Kasunic and Anderson, 2004] and in the European Regional Information Society Association (ERISA). By the term 'barrier' we mean an 'incompatibility' or 'mismatch' which obstructs the sharing and exchanging of information. Three categories of barriers are identified: conceptual, technological and organisational.

- **Interoperability approaches** represents the different ways in which barriers can be removed (integrated, unified, and federated)

[Li et al., 2013] used comparisons between reusability and interoperability, composability and interoperability to show the importance of interoperability. The authors proposed to use models to build interoperability. Thus they decomposed interoperability into:

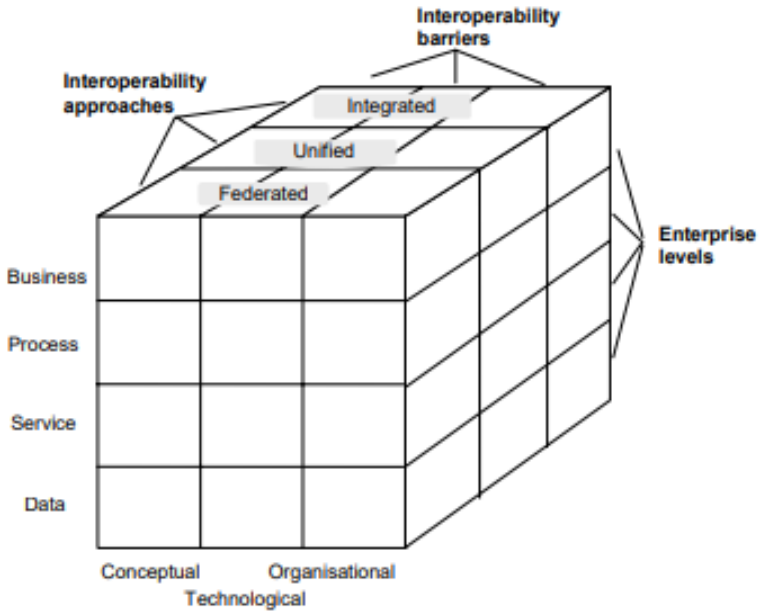

Figure 6: Enterprise interoperability framework - Source: [Chen, 2006]

- technical (or structural) interoperability (communication ports)
- substantive (or semantic) interoperability (contents meaning)

It means that sub-systems can talk to each other and exchange data, but to understand each other correctly and co-simulate effectively substantive interoperability is required.

We assume that interoperability faces two main issues:

- The heterogeneity of the models
- The opening of the simulators

2.4.1 Issues: The heterogeneity of the models

It is very difficult to find all the necessary models we need in a single library. In fact, designers have at their disposal a large choice of models that are described with several approaches, formalisms, and levels of modeling. We can classify the differences between the models according to:

- **temporal natures**: According to time we may classify models into:
  - Continuous models: The variation of the state variables of this type of model is continuous over a finite interval of time. Based on the differential equations, two types of continuous temporal description can be distinguished, [Cellier, 1991]: the ordinary differential equations ODE and the algebraic differential equations DAE
  - Discrete models: Where variables are defined only at specific times. There are models sampled and/or discrete events [Zeigler et al., 2000], but also methods like QSS (Quantized State System) discretizing state variables rather than time [Wetter et al., 2015].
  - hybrid models: having a combination of these two aspects (continuous / discrete)
- **Modeling approaches natures**: Different approaches can be distinguish:
  - Analytical and numerical models are based on physical laws, they usually solve governing flow equations for particular initial and boundary conditions. While empirical models are often derived from the measured system, they have typically been developed using a regression analysis of field observations [Kandelous and Šimnek, 2010]
  - A model where its physical equations are accessible and modifiable is classified as **"white box"**. While a model where only its inputs/outputs are accessible is classified **"black box"**. The combination of the "black box" and the "white box" concepts provides a system called **"gray box"** which is constituted by composition (cf. Figure 7).
  - An explicit model, where the outputs are calculated according to the inputs is classified *"causal"*. While an implicit model where there is no inputs and outputs is classified *"acausal"* (cf. Figure 8). These two approaches redefine models that were defined in 2 as follow:
    - Acausal models are composed of **variables** which expose implicitly changes inside models and **relations** which act as constraints between the values variables take at each instant
    - Causal models are composed of **inputs** which handle data coming from the environment, **outputs** which handle data to be exported to the environment, **variables** and **state variables** which are used to compute observable quantities.

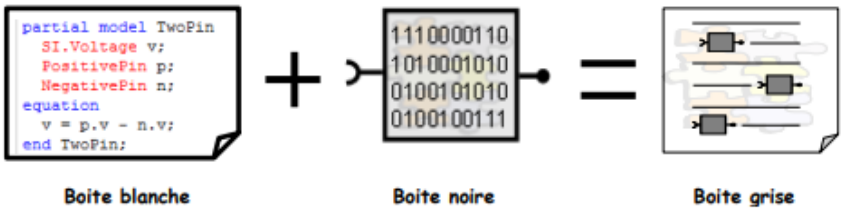

Figure 7: White box/black box/gray box approach - Source: [Gaaloul, 2012]

Figure 8: causal approach (oriented) / acausal approach (unoriented)

2.4.2 Issues: The opening of the simulators

In our case study we can not engage interoperability solutions that ensure an adaptation of the models and a broad compatibility between them. This heterogeneity of models due to a difference in the purposes for which they were designed is inevitable. Instead, we are dealing with the simulators issues and distinguish:

- The multiplicity of simulation tools, because of the lack of communication and the large number of available tools each collaborator use a tool that suits him best. In our work we are dealing with different kind of simulators (Matlab simulink, Powersims, Cooja Contiki see 1.3.1)

- The inadequacy of simulation tools for new needs, experimentally proved in our neOCampus operation, due to the permanent evolution of the simulation tools. Indeed, indispensable tools at a time may be unsuited to current and future situations. As the scientific field is in evolution some of the simulation tools used by some of our collaborators may change, so the need for our co-simulation platform to deal with these changes. Which means that the notion of an open system in which the components are interoperable and easily interchangeable must be brought closer to the development of our complex system.

Capterra `https://www.capterra.com/simulation-software/`, which is a free resource to help find the right software for a business, quotes the majority of existing simulation tools. The Interoperability of heterogeneous simulations is then essential to ensure the coupling between them.

2.5 Existing solutions and interoperability standards

2.5.1 Solutions for interoperability

interoperability is defined by the IEEE as "the capacity of two or more systems or components to exchange information and to use that information that has been exchanged". Different interoperability approaches exist, we can distinguish:

Data interoperability This approach is based on data exchange between programs. This exchange can be made:

- *point-to-point* in the case of a low number of connections and develop a data adapter between each pair of tools
- *centralized standard format* solution that aims instead to federate tools around a "common database". Thus, each tool must develop a single connector to this data format, allowing it to exchange with all those already connected. in the field of building this concept is democratized by the name BIM (Building Information Modeling) [Eastman et al., 2011]. We can mention some existing standards:
  - IFC (Industry Foundation Classes) [Bazjanac and Crawley, 1999]: is a standard initiated by the IAI (International Alliance for Interoperability). Its an object oriented file format, and specific in the industry of construction, it ensures the transfer of data between tools in the different stages of the building life
  - GbXml2 (Green Building XML): this format ensures interoperability of data exchange between the development or design tools used in the building
  - CityGML3: This is an open standard exchange format for storing digital 3D models of cities and landscapes

Data interoperability is necessary for the transfer of information between tools in general, and those simulating particular models. However, this approach represents only a part of the interoperability requirements. In fact, the interoperability of the data is transversal, it is present in parallel with each of the other interoperability approaches presented below:

interoperability based white-box approach This approach is based on the use of a single generic and neutral language for the definition of the different models, known as a white-box approach. This white-box view allows the implementation of a component to be specified in terms of other basic components, providing a hierarchical resolution of a system. This view of a component can be used to show the classes that work together to provide the functionality at the

component interfaces. Ports can be connected to internal classes to show which classes provide the functionality at component interfaces. As example we can cite MODELICA language, described in [Tiller, 2001], allows to model the multi-physics systems (energetic, thermal, aerodynamic, etc.), originally from the field of mechatronics, also very close in the concepts of the VHDL-AMS, `https://fr.wikipedia.org/wiki/VHDL-AMS`, standard derived from microelectronics. Unfortunately we can't apply this approach in our case because of the lack of full knowledge of the physical equations of the models
This kind of interoperability has advantages:

- Strong coupling (acausal): the definition of all trades under a single language in the form of a system of equations ensures a strong coupling in the resolution and allows in particular to deploy quite easily, in a unified language, an approach acausal, as does Modelica.

- Fast computation time: the resolution in a single platform and by a single solver, eliminates the external exchanges and can make the resolution faster under certain conditions (in particular when the models are strongly coupled and require strong interactions of computation)

and disadvantages:

- Dependency of a modeling language: the obligation to re-implementation of all models under a single language is tricky for some models

- Requirement of a great physical knowledge to have the capacity to project different models in a new language

- A single solver for the whole model, so not necessarily adapted to all physics. Can lead to poor conditioning of the overall model

- Low compatibility with other tools

- A single time step, which adapts to the smallest dynamics and induces a longer simulation time

- Lack of confidentiality: the models are written in a Standard format accessible to everyone in a white box approach

In this way the end-to-end flows through system components can be documented. we can give UML as example.

Interoperability based black-box approach This components approach is the opposite of the interoperability based white-box approach. This exchange can be made at the source code level or, a more interesting perspective, at the pre-compiled libraries (static or dynamic), such is the case of the strategy adopted for the exchange of models from TRNSys to Matlab/Simulink to through the dll (Dynamic Link Library) corresponding to the "type" of TRNSYS [REID 2009].The emergence of standards compatible with a wide range of modeling tools was needed, which led to the specification of certain standards such as the FMI [FMI 2010a] seeing their origins in the AUTOSAR standard of the automotive industry.
Thanks to the emergence of standards compatible with a wide range of modeling tools, this approach is very interesting and is used more and more.
The component concept, on which the approach is based, is defined by [Szyperski, 1996] as a unit of composition with contractually specified interfaces and explicit context dependencies only.
The components interoperability has many advantages:

- Independence: autonomous software component, contains its own solver (which is the case of some of the components we're dealing with in neOCampus)
- Easy operation: a simple command allows the simulation of components, with the only knowledge of the inputs / outputs
- Wide range of tools compatible with the use (import and export)
- Confidential: black box approach, no access to model code (which is very important for us and not supported by interoperability of the models)

The disadvantages are limited to the low coupling, and therefore potentially slower simulation, and when a change occurs the need to regenerate software components, and return components to users. We should mention that the coupling is defined as a metric indicating the level of interaction between two or more software components [Akoun and Yonnet, 1984]. The "strong" or "weak" notion of a coupling is related to the information rate exchanged between the coupled components. the definition and the implementation of the strong coupling is delicate. They are often implemented at small granularity levels (within a "component") because these components are often reused in their entirety and the investment can therefore be made profitable.
The weak coupling, in turn, allows more flexibility because of their ease of implementation so involved at a higher level of granularity.
We chose the black-box component approach in order to overcome the limits of the white box approach which consists on redeveloping of all the heterogeneous subsystems in the same language [Kossel et al., 2006]. This imposes that models developped by different designers are transparent and accessible [Allain et al., 2002]. The component approach makes it possible to co-operate prefabricated pieces, perhaps developed at different times, by different people, and possibly with different uses in mind. The main reason is to improve the flexibility, reliability, and reusability of our framework due to the (re)use of software components already tested and validated avoiding risks of robustness. A component is an autonomous deployment entity which encapsulates the software

code showing only its interfaces. An interface can be described as a service abstraction, that defines the operations that the service supports, independently from any particular implementation [Lea and Marlowe, 1995]. Each component should provide the way how it can be generated (plug-out) either from a white box model, or another black box provided by a simulator. We show in (cf. Figure 9) a communication between two software components; componentA (as a piece of software) and componentB (UML notation).

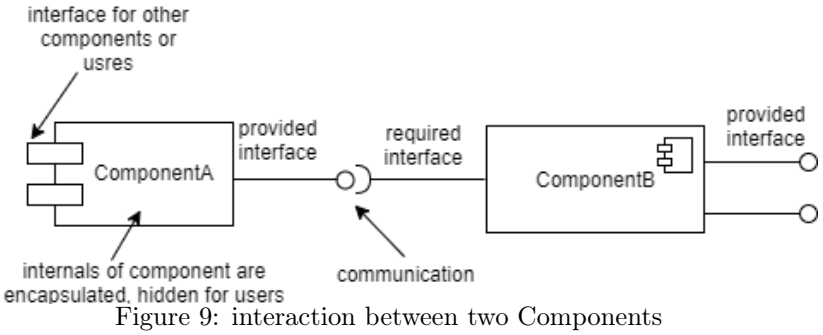

Figure 9: interaction between two Components

Cooperation of data and processes of models This approach, based on [Hensen et al., 2004] studies, focuses on the coupling of programs for a mutual exchange of information in a predefined manner during the simulation. This strategy is also called "co-simulation" and allows more modeling and simulation flexibility than other approaches, since models are independently created and separately simulated in their own tools. In some cases, it is the only possible solution for interoperability, like in our case, when it is difficult to separate the models and their data from their solver. In addition, the other solutions are preferable when possible because co-simulation offers low performance and potential problems of numerical instability. For instance, an effort of generalization and automation to facilitate the implementation of a co-simulation has been realized. This is the BCVTB "Building Controls Virtual Test Bed" [Wetter, 2011], a platform ensuring the coupling of several tools and with which several connections have been developed. This is a standardization of the way of communication: the exchange of information via sockets[5]. Orchestration is provided by the Ptolemy[6] multi-domain environment for the modeling and simulation of heterogeneous systems. A domain in Ptolemy[7] means the set of rules that make it possible to interpret a model including: the mode of time management and the mode of synchronization and communication between models. A very interesting work of standardization of co-simulation actors has been demonstrated in [Consortium et al., 2010].
After an analysis of the existing solutions and the needs of the field in term of interoperability between models and simulation tools, in order to be able to implement a global simulation of the building system, the convergence towards a solution allowing both the exchange of models between the tools as well as the cooperation of the simulators becomes essential. For that we distinguish

[5]software interface with the operating system services, through which a developer will easily and uniformly exploit the services of a network protocol such as TCP/IP
[6]a software environment for design and analysis of heterogeneous systems
[7]WEBPTO: http://ptolemy.eecs.berkeley.edu/ptolemyII/

in the following the different standards of interoperability, before identifying the most suitable interoperability solution for our case study.

2.5.2 Interoperability Standards

We built a table that summarizes the different standards of interoperability cited in order of seniority:

SIMNET: Simulation networking Copies of the same simulator in a common synthetic world [Calvin et al., 1993]	**GDS: Generic Data System** Very different simulations for exchanging global data and putting them in a system of deposit and analysis of common data [Smith, 1996]
DIS: Distributed Interactive Simulation Joins a wide variety of simulations and improves SIMNET (for the family of virtual simulators)	**ALPS: Aggregate Level Simulation Protocol** More dynamic environment providing distributed management functions [Wilson and Weatherly, 1994], Improves GDS (for the family of aggregated simulators)
HLA: High Level Architecture (Which appeared to unify the two families mentioned above) Uses RTI "runtime infrastructure", Supports virtual and interactive simulations (complies with 10 rules, associates your simulation with other people via a specification interface (document describing the service to be called), declaring our objects in an OMT format "Object Model Template "(to unify the description) [MODELING, 1995]	
CCSIL: Command Control Simulation Interface Language A language to be communicated between and within control entities, and small units of virtual platforms generated by computers [Smith, 1996] **SEDRIS: Synthetic Environment Data Representation & Interchange Specification** Infrastructure technology that enables information technology applications to express, understand, share and reuse environmental data [Foley et al., 1998]	
MUSE: Multiple Unified Simulation Environment The MUSE component. Unified Model for Energy Devices and Systems, (MUSE) is a support for a new paradigm of models. This standard offers the possibility to communicate models of varied nature [Perhinschi et al., 2010]	
FMI: Functional Mockup Interface Standard interface for the solution of coupled time dependent systems, consisting of continuous or discrete time subsystems. It provides interfaces between master and slaves and addresses both data exchange and algorithmic problems. Simple and sophisticated master algorithms are supported. However, the master algorithm itself is not part of FMI for Co-Simulation and should be defined [Consortium et al., 2010] [Enge-Rosenblatt et al., 2011]	

Table 1: Simulation Interoperability Standards

Here is some interoperability standards as they evolved we mention that **SIMNET** was a simulation protocol. As U.S. Department of Defense (DoD) built all these (military) network simulators they realized they needed some kind of standard interoperability protocol to tell each other where they are on a battelfield and what they're doing and so SIMNET was also the name of a protocol that support those network operations, that Was followed closer after by Distributed Interactive Simulation (**DIS**). In this last one, DIS, they tried to stay away from

tank's centric way of viewing the battlefield, they tried to include objects and events that tanks wouldn't engage in, but helicopters jets and ships may engage in, tring to make it more generic more jointly, so that they could all use the same protocol rather than relying on an army protocol which was what SIMNET primarily was. DIS is a virtual reality system interconnected over long distance networks that share information through individual and cooperative interactions. One of the most important aspects of these applications is the ability of each site to manage in real time the states of all participating entities (orientation and position) in the exercise. Simulation state information is encoded in formatted messages, known as protocol data units (PDUs[8]) and exchanged between hosts using existing transport layer protocols, including multicast[9], though broadcast User Datagram Protocol (UDP[10]) is also supported.

At the same time there was something called Generic Data System (**GDS**) which was developed by the United States Air Forces in Europe (USAFE) Warrior Preparation Center in Germany. The USAFE owned the copy of each of these services major staff terrain wargames and they were trying to get them all together, so they were developing a protocol of their own. The joint training confederation was being formed up kind of parallel to that. They invented a protocol called Aggregate Level Simulation Protocol (**ALSP**). So we ended up with one standard **DIS made for virtual simulators**, and one **ALSP made for aggregate simulators**, which wasn't good enough for unification.

The **HLA** came out using a thing called RTI (Runtime Infrastructure[11]), which is a single software package, and meant to support both virtual level simulations and constructive level simulations and waregames that are used for analysis. A system is seen as a federation regrouping several federates[12], communicating by mechanisms of publications/subscription see (cf. Figure 10). This decomposition by federates makes it possible to combine several types of components such as simulation models, functional codes (in C ++ or Java) or hardware devices. The main benefit of HLA is interoperability and reuse. The HLA standard is based on the message-passing paradigm and specifically defines high-level services for the management of time. The implemented algorithms ensure that there is no causal loop in HLA simulations. Indeed, each federate declares a lookahead that defines a time horizon during which it will not emit any message. The smaller the lookahead, the more message exchanges there will be. When the lookahed is zero, algorithms ensure the absence of deadlocks.
CERTI is an Open Source HLA RTI developed by the French Aerospace Lab ONERA [Noulard et al., 2009]. CERTI supports HLA 1.3 specifications (C++ and Java) and partial IEEE 1516-v2000 and IEEE 1516-v2010 (C++).
To be HLA compliant: 3 guidelines to follow:

- The first one is the set of HLA rules to follow, five rules describe how your simulation which is called a federate in HLA environment will operate. Five other rules describe how a

[8] is information that is transmitted as a single unit among peer entities of a computer network
[9] group communication where data transmission is addressed to a group of destination computers simultaneously. Multicast can be one-to-many or many-to-many distribution
[10] is one of the core members of the Internet protocol suite, allowing computer applications to send messages, in this case referred to as datagrams, to other hosts on an Internet Protocol (IP) network
[11] underlying middleware whose role is to ensure the smooth running of the simulation
[12] simulation entities performing a series of calculations interconnected by a Run-Time Infrastructure (RTI)

federation as a family will operate. These rules are binding you into HLA as the only way to exchange data, they're making sure that you're not using any backdoors. For example, one of the rules specifies that all data exchanges between federates must pass through the RTI.

- The second one is to attach your simulation (federate) to other simulations via a thing called HLA Interface specifications, which is a document describing what services you will call when you want to do something that affects other simulators. We should mention that a specification itself is not a software its just a document that describes what that Application Programming Interface (API) will be that connect you to the rest of the federation. For example, the interface describes how a federate can join or create a federation.

- The third one is to describe your objects in Object Model Template (OMT) format. an object model template (based on the OMT standard (Department of Defense (DoD) Specifications, 1998b)) providing a common environment for descriptions of communications between HLA simulations. For each federation, a Federation Object Model (FOM) describes shared objects, attributes, and interactions.

HLA time management services enable deterministic and reproducible distributed simulations. Each federate has a logical time and the RTI ensures the coordination of federates by consistently advancing the logical times of each federate. A possible implementation of synchronization mechanisms is based on the Chandy and Misra algorithm (Chandy, Misra, 1979). Depending on performance requirements, other algorithms may be used. Logical time is used to ensure that federates observe events in the same order (Fujimoto, 1996). The logical time is equivalent to the simulation time in the classical literature on discrete event simulation.

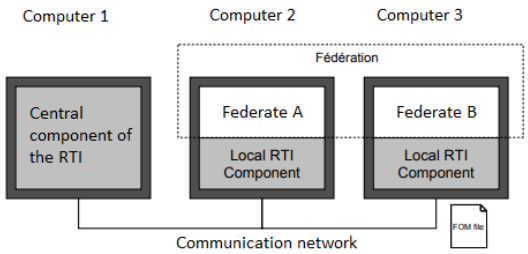

Figure 10: High level view, logic of a HLA Federation execution [Zacharewicz, 2006]

Command Control Simulation Interface Language (**CCSIL**), which is very similar to DIS network package, tries to capture information that needs to be exchanged by two brains or Artificial Intelligent (AI) models, trying to say words in a way that can be understood by an AI computer.

SEDRIS has followed up, it essentialy tried to define one way to describe terrain data, one way to describe envirenmental data. DIS didn't give us mecanisms to create compatibility between databases (this came later under SEDRIS). Making it possible to publish informations about the changes in the battlefield and deliver them to the other simulators fast enough so they don't see the scene freeze up. ALSP has lot of similarities with DIS, in addition it controls time

both in moving forward faster and slower and back in time and starting over.

The **MUSE** component see (cf. Figure 11), Unified Model for Energy Devices and Systems, stemming from the ANR PLUMES project, is a support for a new paradigm of models. The successive paradigms in computing: which are objects, then components and services [Donsez, 2006], offer innovative solutions to the domains of model-based engineering, which can be seen as objects, components, or services.

As the following figure (cf. Figure 12) illustrates, a hierarchical structure with 3 levels of gran-

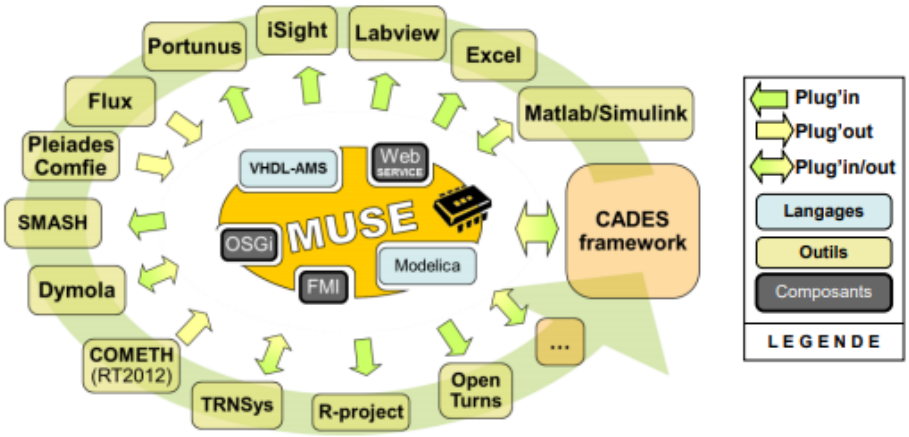

Figure 11: The communication capability MUSE. Source: (MUSE)

ularity can be defined in terms of reuse. The concept of services aggregates that of components, aggregating itself that of objects, each layer offering its properties in terms of compromise of dynamics and coupling force. The lower the coupling, the stronger the dynamics, i.e the couplings can be changed more quickly and easily.

Compared to the service-oriented approach, component-oriented programming offers the ability to

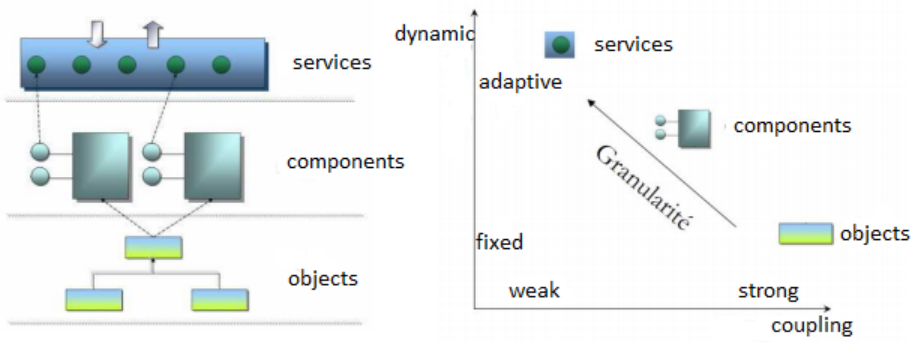

Figure 12: Hierarchical architecture of object/component/service, and dynamic/coupling compromise. - Source: [Delinchant et al., 2012]

deploy the model among its different users. To do this, the models must embed in the component all its dependencies thus making it autonomous. When this autonomy is not possible, or too complex to obtain, the service-oriented approach takes over. The main difference in this case lies in the fact that the execution of the model is delocalized on the "cloud".
MUSE sprung from a previous standard, the **ICAR** component (Interface for Component Architecture), ICAR has characteristics that are the basis for defining MUSE:

- Black box vision whose component's code is unaccessible
- Ability to deal with multi-business using the facet concept
- Hierarchical Composition of instances of arbitrary industrial component models yielding new (compound) components with specified capabilities and requirements which can themselves be composed to yield higher level components
- Ability to generate components from a white box description (including Modelica)
- OSGi[13] Compatibility

Muse make it possible then for each model developed for a specific simulation tool to be used by other simulation tools.

The Functional Mockup Interface (**FMI**) is an interface type that seems to be evolving as the standard for coupling physical models and simulators. The FMI offers a low-level interface for two purposes depending on the needs to be achieved:

- model exchange which exposes a compiled numerical model to a solver/simulator with a standard interface to initialize the states, execute a time step, determine derivatives, etc. The model is wrapped into such an interface resulting into a Functional Mockup Unit (FMU). This last could be plug into a bigger system model to verify if that particular component interoperates well with the remaining components in the intended system
- co-simulation which goes one step further and also packs the solver into the FMU. This FMU acts as a co-simulation slave, orchestrated by a master algorithm, which cares for synchronization, variable exchange, etc.

[13]standard that would allow automatic management of dependencies between components, deployment on embedded platforms (control command), etc. For example, it is the basis of the Eclipse modular platform, and therefore of the Papyrus UML[14] tool

2.5.3 Discussion

Among the different standards of simulation interoperability cited, the two most well known are HLA (High-Level Architecture) and FMI. In HLA federates (components) are connected to a central Run-Time Infrastructure (RTI). In **FMI**, components are FMUs (Functional Mock-up Unit, slaves) see (cf. Figure 13), and a master must be written to transfer data between FMUs and order the execution of a time step. To avoid the obligation imposed in HLA to use RTI libraries from the same provider and usually of the same version in order for applications to interoperate, we chose the standard FMI [Blochwitz et al., 2011].
[Garro and Falcone, 2015] investigates how to combine HLA and FMI from two different perspectives: (1) HLA for FMI and (2) FMI for HLA. With reference to the HLA for FMI perspective, some possible extensions to FMI to include HLA features are proposed.
FMI uses a master-slave architecture, described in subsection 2.1. It is a standard which minimizes the customization of the exported model. It is an independent tool that facilitates the exchange of models between different tools and which therefore reduces the effort of integration by proposing approaches that are specific to it.

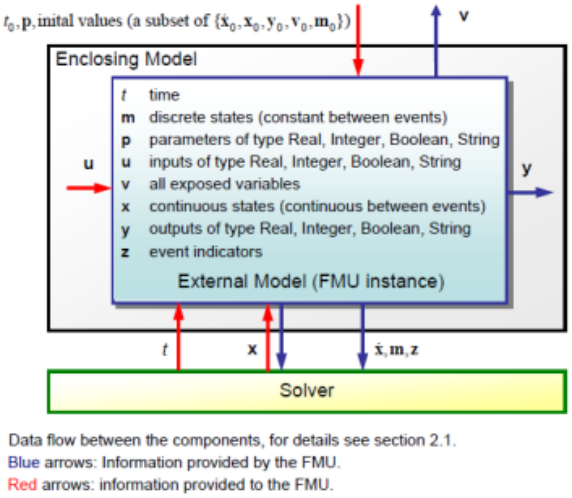

Figure 13: FMU for the dynamic simulation

Our solution is based on the combination of both data and components solutions of interoperability, and inspired from the decomposition of [Li et al., 2013] but instead of using models, which imposes an important customization of the exported model, we take advantage of a known simulation standard, FMI (Functional Mock up Interface), described in the next section, to build **the structural part**. Our **semantic interoperability** will be based on data mediation in a first contribution, and on AMAS "Adaptive Multi-Agent Systems" (described later) in a second one.

3 Interoperability layers (Structural and Semantic)

3.1 Structural interoperability based FMI

3.1.1 FMI: Functional Mockup Interface

FMI, see (cf. Figure 14), is a standard interface for the solution of coupled time dependent systems, consisting of continuous or discrete time subsystems. It provides interfaces between master and slaves and addresses both data exchange and algorithmic problems. The development of FMI was initiated and organized by Daimler AG within the ITEA2 project MODELISAR[15] [Consortium et al., 2010]. The primary goal is to support the exchange of simulation models between suppliers and OEMs even if a large variety of different tools are used. The FMI was developed in a close collaboration between simulation tool vendors and research institutes [Blochwitz et al., 2011]. Simple and sophisticated master algorithms are supported [Enge-Rosenblatt et al., 2011].

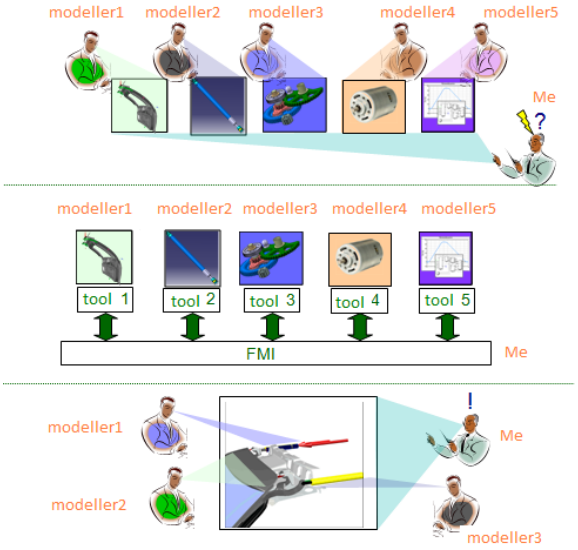

Figure 14: Overview of FMI

FMI supports different working modes, particularly:

- model exchange: when a modeling environment can generate C code of a dynamic system model, in form of an input/output block, that can be utilized by other modeling and simulation environments. The models to be treated can be large for usage in offline or online simulation and in embedded control systems on microprocessors.

- co-simulation: when an interface standard is provided for coupling simulation tools in a co-simulation environment. The data exchange between subsystems is restricted to discrete

[15] http://www.modelisar.org

communication points. In the time between two communication points, the subsystems are solved independently from each other by their individual solver. Master algorithms control the data exchange between subsystems and the synchronization of all slave simulation solvers (slaves). The interface allows standard, as well as advanced master algorithms, e.g. the usage of variable communication step sizes, higher order signal extrapolation, and error control [Blochwitz et al., 2011]. see (cf. Figure 15).

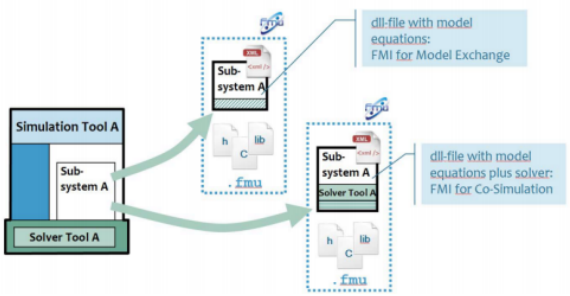

Figure 15: Models Exchange vs cosimulation. Source: `https://fmi-standard.org/`

3.1.2 Functional Mockup Units (FMU)

A Functional Mockup Unit (FMU), according to [Blochwitz et al., 2011], is a component which implements the FMI interface. It consists of one zip-file with extension ".fmu" containing all necessary components to utilize the FMU:

- An XML-file contains the definition of all variables of the FMU that are exposed to the environment in which the FMU shall be used, as well as other model information. It is then possible to run the FMU on a target system without this information, i.e., with no unnecessary overhead. For FMI-for-Co-Simulation, all information about the “slaves”, which is relevant for the communication in the co-simulation environment is provided in a slave specific XML-file. In particular, this includes a set of capability flags to characterize the ability of the slave to support advanced master algorithms, e.g. the usage of variable communication step sizes, higher order signal extrapolation, or others.

- For the FMI-for-Model-Exchange case, all needed model equations are provided with a small set of easy to use C-functions. These C-functions can either be provided in source and/or binary form. Binary forms for different platforms can be included in the same model zip-file. For the FMI-for-Co-Simulation case, also a small set of easy to use C-functions are provided in source and/or binary form to initiate a communication with a simulation tool, to compute a communication time step, and to perform the data exchange at the communication points.

- Further data can be included in the FMU zip-file, especially a model icon (bitmap file), documentation files, maps and tables needed by the model, and/or all object libraries or

DLLs that are utilized.

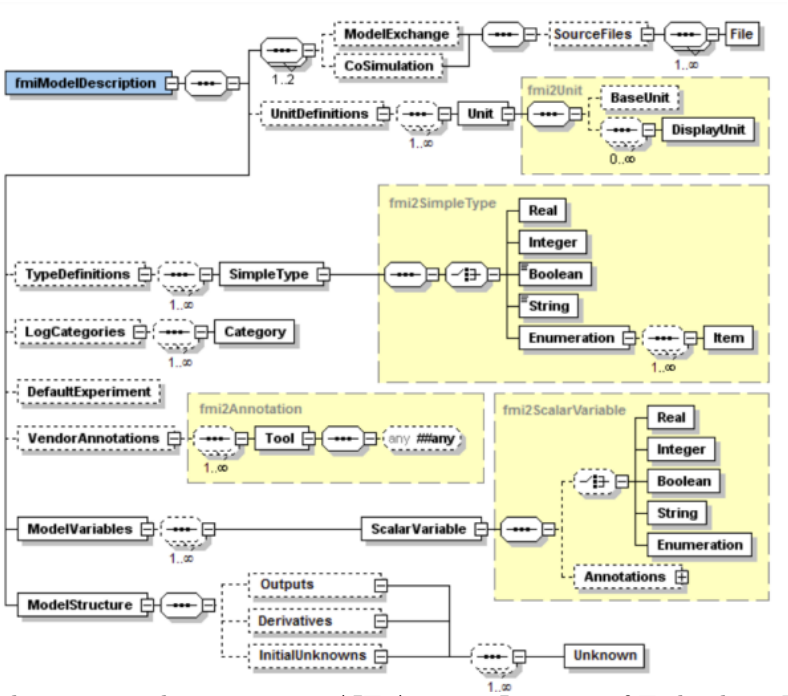

Figure 16: FMI description schema. source: AIT Austrian Institute of Technology, ERIGrid JRA2 Workshop

3.1.3 FMI description schema

An XML-file called "modelDescription.xml" store the informations about a model and a co-simulation setup that is not needed during execution. Every tool can then use its favorite programming language (e.g. C, C++, C, Java, Python) to read this XML-file, reducing the overhead, both in terms of simulation efficiency and memory. A representation of the FMI description schema is shown in (cf. Figure 16). An important part of the FMI for co-simulation is the definition of capability flags (to define the capabilities that the co-simulation slave supports) interpreted by the master to select a co-simulation algorithm which is supported by all connected slaves. as you can see in (cf. Figure 17).

3.1.4 C-interface

To increase flexibility in use and portability to virtually any platform, FMI provides fundamental functionalities in the form of a C interface. Three header files that define the C-types and –interfaces can be distinguished:

- The "fmiTypesPlatform.h" file which contains all definitions that depend on the target platform, and which must be used both by the FMU and by the target simulator, see (cf. Figure 18)

- The "fmiFunctionTypes.h" file which contains typedef definitions of all function prototypes of an FMU, used to type-cast the function pointers to the respective function definition when dynamically loading the DLL or shared object of an FMU

- The "fmiFunctions.h" file which includes the two header files from above, and contains the function prototypes of an FMU that can be accessed in simulation environments

Figure 17: Capability flags of FMI for Co-Simulation

```
#define fmiTypesPlatform "standard32"
#define fmiTrue  1
#define fmiFalse 0
#define fmiUndefinedValueReference
                (fmiValueReference)(-1)

typedef void*         fmiComponent;
typedef void*  fmiComponentEnvironment;
typedef void*         fmiFMUState;
typedef unsigned int fmiValueReference;
typedef double        fmiReal   ;
typedef int           fmiInteger;
typedef char          fmiBoolean;
typedef const char*   fmiString ;
typedef char          fmiByte;
```

Figure 18: The "fmiTypesPlatform" header file, source: [Blochwitz et al., 2012]

3.1.5 FMI for Model Exchange (ME)

In this mode, the objective is to generate, from modeling (for example Modelica) executable code that can be used by other modeling and simulation environments. The models are described by differential equations, algebraic and discrete equations including various types of events. The executable models are a priori usable in use online or offline, or in a more constrained context (embedded, micro-controller, etc.). It is also possible to use multiple instances of the same model and to connect multiple models hierarchically. It should be noted that the independence of the models from the target simulators is ensured by the absence of a header file specifically linked to a given simulator. The FMU model distribution is made in the form of an archive.

3.1.6 FMI for Cosimulation (CS)

The goal is to provide a standard interface for coupling two or more simulation tools into a co-simulation environment. In this context, the communication between the subsystems is reduced to discrete communication points, between which the subsystems are treated independently by their respective solvers. A master algorithm controls data exchanges between subsystems and synchronization of all slave solvers.

In the version 2.0 of FMI, one FMU can implement both interfaces (model exchange, co-simulation) at the same time. The presence of the "ModelExchange" or "CoSimulation" elements in the XML-description indicates which interface is implemented. Which interface is used by the environment is decided by calling the appropriate instantiation function (fmiInstantiateModel or fmiInstantiateSlave).

3.1.7 Enumerations of variables

We distinguish 2 categories of FMU variables:

- The "causality" of the variable:
  - parameter: An independent variable that must be constant during simulation
  - input: The variable value can be provided from another model
  - output: The variable value can be used by another model. The algebraic relationship to the inputs is defined in element ModelStructure
  - local: Local variable that is calculated from other variables. It is not allowed to use the variable value in another model
- The "variability"[16] of the variable:
  - constant: The value of the variable never changes
  - fixed: The value of the variable is fixed after initialization
  - tunable: The value of the variable is constant between externally triggered events due to changing variables with causality = "parameter" or "input". Tuning a parameter consists of:
    - Stop the simulation at an event instant (usually, a step event, in other words after a successful integration step)
    - Change the values of the tunable parameters
    - Compute all parameters that depend on the tunable parameters
    - Resume the simulation using as initial values the current values of all variables and the new values of the parameters
  - discrete: The value of the variable is constant between internal events (= time, state, step events defined implicitly in the FMU)
  - continuous: No restrictions on value changes

[16] the time instants when a variable can change its value

3.1.8 FMI implementation

The internal FMU state, consisting of the values of the continuous states, discrete states, iteration variables, parameter values, input values, file identifiers and FMU internal status information, can be copied and the pointer to this copy is returned to the environment and can be set as current FMU state, in order to continue the simulation from it.
Moreover the FMU state can be serialized and copied into a byte vector (e.g. to perform an expensive steady-state initialization), and store this vector on file. Whenever needed, the byte vector can be loaded from file, can be deserialized and the simulation can be restarted from this FMU state.

The use of FMI can be summarized in 4 steps:

- **The design step**: The package of the model of simulation in one component **FMU** embedding the modeling, and transformation (publication of the FMU which contains a xml file and the model code or its file). The main challenge is to fill-up the gap between the semantics of FMI and the semantics of the source formalism of the various calculation models (state machines, discrete event, data flow or timed automata) [Tripakis, 2015].

- **The composition step**: The model of the subsystem is joined to the complex system by establishing the connection graph of the simulation components.

- **The deployment step**: The FMUs are made available to the slave simulators. This can be made offline (manually by the user) or online (automatically by the master and where the user specifies in which network the instances of the FMUs are transferred).

- **The simulation step**: The master is responsible for the life cycle of FMUs instances during the execution of the simulation.

3.2 Data mediation based on semantical interoperability

As the interoperability is the ability to share information between systems and applications in meaningful ways. While most system engineering or system applications stop at the structural level, assuming that if you can read it you're going to understand it. Additional level of interoperability and the next that really matters for us is a semantic one which needs common information model to be defined for exchanging the meaning of information. Then the content of the information exchanged is unambiguously defined.

3.2.1 Data mediation

If every system always used the same data to represent the same information – identical names, structure, and representations – then the data interoperability problem would go away [Renner et al., 1996]. Unfortunately it's not usually the case. Furthermore the construction and

maintenance of a single, integrated standard data model is difficult. We should expect many models, each covering a single functional domain, which leads us to data interoperability issues wherever systems communicate across the boundaries of separate models.

The term semantic interoperability was first coined by Sandra Heiler [Heiler, 1995], and tries to ensure that the exchange of services and data among components in large-scale distributed systems makes sense. This semantic interoperability level should go beyond operational semantics or behavioral specifications of components [Hernández et al.]. Agreements on names and meaning of shared concepts are also needed, as well as context dependencies, quality of service information, non-functional requirements, etc. [Heiler, 1995].

Depending on the problem or the properties that need to be solved many proposals to endow components with behavioral specifications: temporal logic [Han, 2000], pre/post conditions [Zaremski and Wing, 1995], process algebras [Bernardo et al., 2002], petri nets [Murata, 1989], refinement calculus [Chouali et al., 2006], etc. At this behavioral level we usually distinguish two concepts: the compatibility, where the behavior provided by a component and the one expected by its client component are accordant (a "design by contract" discipline was born), and the substitutability known as the behavioral subtyping, introduced by [America, 1990] as the consistency of the behavior of subclass instances with that of superclass instances.

In [Yellin and Strom, 1997] the protocol level of interoperability was identified as being above the structural and signature level (i.e., the names and profiles (parameter types and return values) of the components operations), dealing with the partial order between the components exchanged methods, and the blocking conditions that rule the availability of the components services. Unfortunetly this protocol level doesn't cover all the semantic aspects of components. Since each person seems to have a clear idea of what semantics is, which makes semantics a broad, fuzzy concept [Vallecillo et al., 2000].

In our work, we distinguished 3 ways to achieve the semantical interoperability level, either by data mediation, by establishment of an Ontology, or by using dynamic data mediation based on Artificial Intelligences.

As the prime objective of an ontology is to model a set of knowledge in a given domain [Noy et al., 2001], and as the complexity of our work is irregular and evolve in time. We noticed that the degrees of collaboration between actors needed for that purpose, and the risk to loose some properties like openness is high. We choose firstly to approach the semantical interoperability using data mediation, (cf. Figure 19), which will be introduced below, then we proposed to adapt our approach by using adaptive multi-agents systems that will be detailed later in the chapter 2.

The concept of mediation [Wiederhold, 1995] appears when we want to get components to work together when they are functionally compatible (at high enough level of abstraction, the provided and required services by two components are equivalent), but not necessarily signature and/or protocol compatible.
The purpose is to build some adaptors or mediators able to adapt their interfaces, bridging their incompatibilities. Nevertheless, [Yellin and Strom, 1997] shows the difficulty of their automated

construction right from the description of the interfaces of the original components.

The information exchange between a source and receiver system is handled by the mediator[17], beginning with a query from the receiver's schema, we first translate it into the equivalent query against the source schema. Then, we execute the source query and translate the retrieved source data into the receiver's format. So the mediator acts as a semantic gateway between the systems, making it possible for the receiver to view the source as an extension of its own database, without concern for the differences in names and representations of data.
The first approach uses a model that describes the information shared by taking their semantics

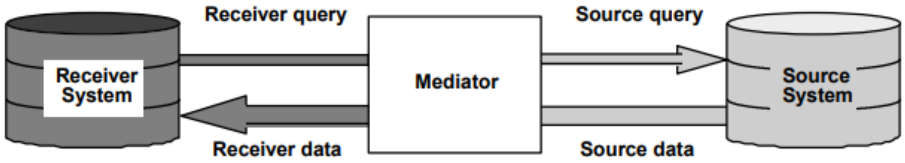

Figure 19: data mediation

into account in the form of contexts of use. The mediation model leads to define three types of integration rules:

- constraint rules that reduce the objects to be considered according to predicates,
- merge rules that aggregate instances of classes of similar,
- and join rules that combine information from multiple object classes based on one or more common properties.

3.2.2 Types of mediation

We distinguish two types of mediation:

- Schema mediation [Halevy et al., 2003] which provides better extensibility and often better scalability (object interfaces, rule-based language).
- Context mediation [Roy et al., 2007] seeks to discover data that is semantically close, it is able to locate and adapt information to ensure complete transparency. We can therefore take advantage of the robustness of the mediation schema approach and combine it with the semantic approximation techniques of context mediation. This semantic mediation will be used to correlate, aggregate and dispatch data with respect to the control that we want to enforce on data produced or consumed.

[17]The mediator generates data translations from descriptions of the data, which are: written using a formal language and a common vocabulary, then adequate for the mediation that is required

Different types of semantic knowledge can be represented within a Context Interchange system in order to provide the knowledge needed for context mediation. A Context Mediator compares the contexts of the data sources and receivers to determine if semantic conflicts exist, and if so, what conversions need to take place to resolve them. This is referred to as conflict detection and the information detailing the conflicts are presented in a conflict table. Then an inserting of the appropriate conversion operations identified in the conflict table need to be done. Finally, the intermediate answers obtained from component systems must be merged together and converted to the context expected by the receiver.
A common vocabulary would be appreciated to ease the transformation of data, so that data wihin a given context (of one component) may be meaningful in another context. we can make assertions for example that "dates" are reported using different formats ("DD/MM/YY" and "MM/DD/YY") according to each context. For these assertions to be meaningfully compared, both contexts must agree that "date" refers to a day in the Julian calendar as opposed to say, a meeting between a boy and a girl; furthermore, they must also mutually agree that "MM", "DD", and "YY" refer to month, day and year respectively.

For example, one component may produce data with meaning T0, while another may consume data with T1. It may be that there is no direct T0T1 bridge, but there are separated T0T' and T'T1 bridges. A mediator is required to assemble the bridges to complete the T0T1 translation. Our solution is based on formal interface descriptions. When a simple component has an input or output event as a kind of interaction, our interface description will list those events including their (typed) parameters. It could be done automatically if a generator tool, based on language mapping, processes an interface description and produces a proxy (environment side stub) and a driver (component-side stub) for the component. Proxy and driver communicate through a mediation channel, using a protocol for message exchange see (cf. Figure 20).

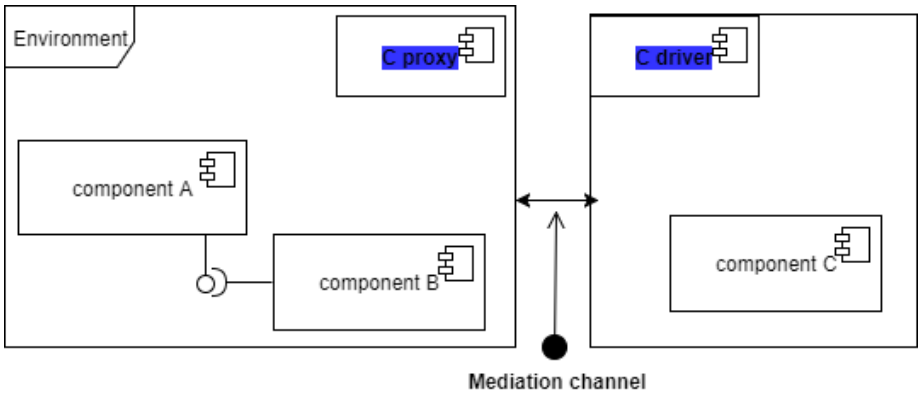

Figure 20: components mediation

A method of federation as a mediation strategy may also be used allowing the components to collaborate together, while allowing them to retain their specificity and their heterogeneity and above all to rely on what exists in the different components. The federation approach seemed to be attractive for our complex system because it offers an opportunity to integrate components in their current state. However, this management of the heterogeneity of the different components has an impact on the architecture; the structure based on point-to-point exchanges would not be

suitable. Indeed, managing the traceability of exchanges between components and the time needed to set up such an architecture could not be satisfactory in the end. Similarly, the distribution of competences and the autonomy of components are two factors against over-centralized control in which a control system would have to assume a hierarchical responsibility covering in itself the expected operations.
The mediation concept is a solution between these two ends. It is a particular variation of the mediation architecture promoted by Wiederhold [Wiederhold, 1992] in a context of information retrieval, with the idea of federating distributed information resources. This approach uses a model that describes the information shared by taking their semantics into account in the form of contexts of use. The Schema mediation mentioned earlier is a direct extension of the federated approach with better extensibility and often better scalability (object interfaces, rule-based language). The use of semantic (web) technologies has been reported in the literature [Schenk and Staab, 2008], e.g., RDF (Resource Description Framework), OWL (Web Ontology Language), SPARQL (SPARQL Protocol and RDF Query Language), Alignments, should contribute to the reusability, interoperability and evolutivity of data and their description. The platform will allow stakeholders to express, ahead of use, their constraints with respect to the control that they want to enforce on data they produce or consume. These constraints will be used for opportunistically negociate data exchange on the platform.

The combination of the Structural interoperability and the semantical one, using the components and data interoperability solutions as described in 2.5.1, allows us to realize a co-simulation. This last will be described in more details in the next section

As we want to find a way of enabling experts in different disciplines, who are part of neOCampus operation, to collaborate more efficiently in the development of our complex systems. Two possible solutions can be distinguished in this context: formal integration and coupling of simulators.

3.2.3 Formal integration

The Formal integration involves using a single formalism for all models. To obtain this homogeneity two solutions are conceivable. Either by translating each model into the same formalism, which may be different from all those already used or not. Or by encapsulating the original formalisms by developing translation interfaces in a common formalism (cf. Figure 21).

DEVS Discrete EVent System Specification, is the formalism -formal model description- that emerges most in the literature [Zeigler et al., 2000], [Vangheluwe et al., 2000], [Zeigler, 2014]. It's the most general formalism for discrete event models, and can also integrate all other types of formalisms, generic to describe all scheduling policies from the execution point of view, and can be used as a model of execution to evolve a heterogeneous co-simulation.
As example, the Agents & Artefacts for Multi-simulation (AA4MM) platform of co-simulation [Camus et al., 2012], using DEVS as execution template. AA4MM is a co-simulation meta-model relying on agents and artefacts (the extension of multi-agents systems described later), it:

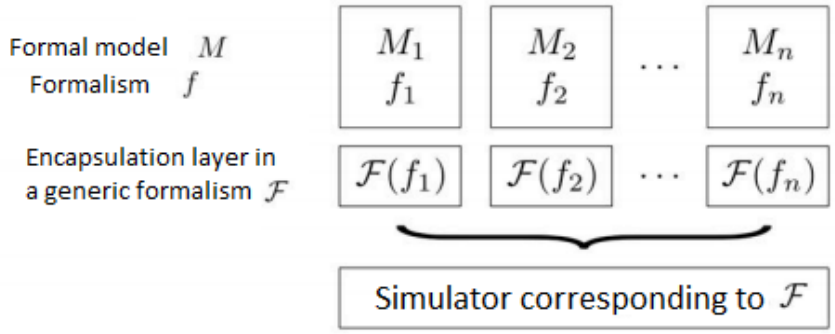

Figure 21: Encapsulation of formalisms. Source: [Siebert, 2011]

- proposes a decentralized approach to co-simulation where the coordination model intends to decentralize the simulators interactions
- relies on DEVS algorithms whose mathematical proof was given that they could not generate deadlocks or break the causality constraints
- co-simulate heterogeneous models and simulators
- AA4MM is described[Camus et al., 2015] by three ontological levels:
  - the paradigmatic level describing the concepts
  - the generic level describing their specializations
  - and their specialization at the domain level

After describing the interoperability standards and focusing on the use of FMI in order to achieve the structural interoperability level. FMI for co-simulation allows coupling of separate tools as such; in this case the FMU consists of an FMI wrapper around the slave tool, which on its turn contains the model of interest and a solver. The former bears the additional advantage that it exempts the user for possessing a dedicated license for the slave simulator. Coupled simulations or co-simulations aim to fulfill the functionality needs by modeling multi-domain systems across multiple simulation tools, while acting as one integral simulation platform that addresses the study. As we want to maintain the original structure of each model, instead of using a single formalism for all models, and simulate them together and avoid the expensive translation process described above. We choose the second solution which is the coupling of simulators as explained in the next chapter.

CHAPTER III: Coupling of heterogeneous simulations using Co-simulation

1 Co-simulation (*co-operative simulation*)

Co-simulation is defined in [Hessel et al., 1999] as the coupling of several simulation tools where each tool handles part of a modular problem where data exchange is restricted to discrete communication points and where subsystems are resolved independently between these points. This allows each designer to interact with the complex system in order to retain its business expertise and continue to use its own digital tools (cf. Figure 22)..

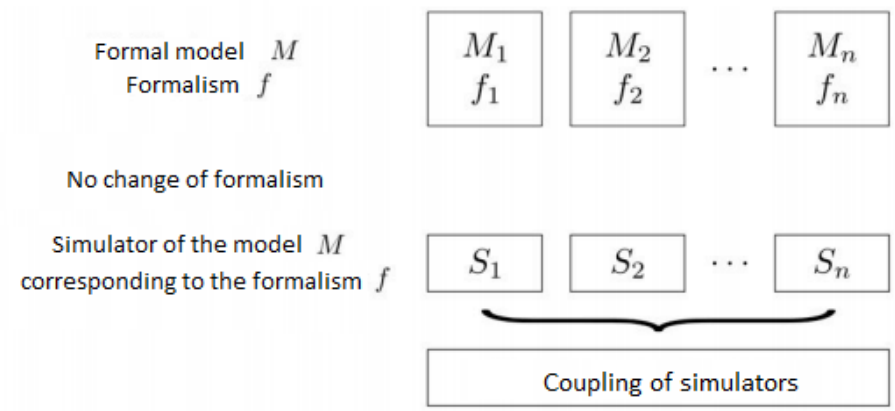

Figure 22: Co-simulation of heterogeneous models. Source: [Siebert, 2011]

For our work we define co-simulation as a set of interacting simulation units, each one needs *Input* data and produces *Output* data. A simulation unit is the composition of a simulator (solver) with a dynamic system (model of a real environment characterized by a state and a notion of evolution rules). The outputs of a simulation unit might be the inputs of others. The progress of the simulated time in each simulation unit is controlled by an orchestrator, also called a master algorithm. This last one, based upon a co-simulation scenario, moves data from outputs to inputs. Therefore, the matching process between the inputs and outputs is difficult especially in open cyberphysical environments where components[18] join and leave the co-simulation on the fly, like in the *neOCampus* operation of the University of Toulouse III - Paul Sabatier [Gleizes et al., 2017]. we detail in the next chapter the various concepts related to the implementation of a co-simulation platform, starting with the synchronization models of exchanges between simulators.

[18] A component is an autonomous deployment entity which encapsulates the software code showing only its interfaces

2 Synchronisation models

Synchronization is an essential element of co-simulation. One of his main goals is to define how data will be exchanged between models. We will focus on the two main synchronization models: the master-slave distributed model and the bus-based distributed model.

2.1 Master-slave distributed architecture

The master-slave model (cf. Figure 23), comprising a master simulation and one or more slave simulations. In this case, the slave simulators are executed using procedure calls, which results in an inability to execute them simultaneously.
Another great difficulty comes from the integration of time [Yoo and Choi, 1997] which is different between embedded software systems, hardware and the surrounding environment.

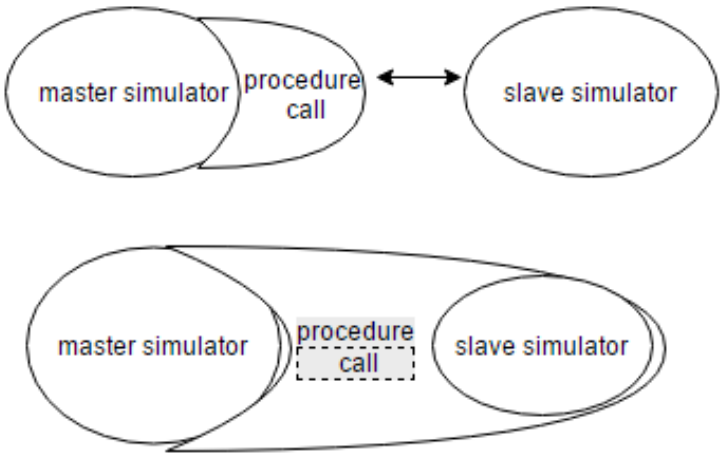

Figure 23: Example of a master-slave co-simulation platform

A slave simulator functions as a black box which is described in 2.4.1, and must present programming interfaces that define its inputs and outputs. We used FMI to realize the structural interoperability and explained that it follows a master slave architecture 3.1. We mention that those FMUs (slaves), need a master to coordinate the overall simulation and transfer data between them. As the master is not part of FMI, it should be implemented by the user of the standard, either with simple or advanced algorithms which can be applied depending on the properties of the involved slave simulators.
While using FMI for Co-Simulation two parts should be distinguished:

- Co-Simulation Interface: set of C functions for controlling the slaves and for data exchange of input and output values as well as status information

- Co-Simulation Description Schema: structure and content of an XML file is defined. This slave specific XML file contains "static" information about the model (input/output variables, parameters, etc.) and the solver/simulator (capabilities, etc.)

Slaves show their ability to support advanced master algorithms which use variable communication step sizes, higher order signal extrapolation, thanks to the capability flags in their XML files. The master is able to import an FMU by first reading the model description XML file contained in the zip file, while the implementation is hided and the intellectual property is protected.
The master is configured by a simple text file. Keywords are used to start and stop time, step size, coupling algorithm, error tolerance etc. The coupled FMUs with their paths should appear within the configuration file. The graph of the simulator coupling has to be supplied by an incidence matrix and information about the priority of the slaves as well as occurring cycles [19].
A simulation tool [20] can be coupled if:

- it can give a time value tci ($tstart <= tci <= tstop$)
- the simulation can be interrupted when arrived to tci (communication point)
- When interrupted the simulation tool can receive values $Utci$ and send values $Ytci$
- When interrupted the simulation tool can take another time value $tc(i+1)$ (to simulate a sub-interval $[tci\ ,\ tc(i+1)]$)

While co-simulating we distinguish 3 phases:

- Initialization phase when all simulation tools are prepared for starting the co-simulation
- Simulation phase where the slaves are forced by the master to simulate the time interval from start time to stop time by stepwise solving master subintervals which are also called communication steps
- Closing phase when the master stops the complete simulation and is responsible for proper memory deallocation, terminating and resetting or shutting down the slaves

The interface between the master and slave should be able to transfer:

[19] A cycle is a path in a graph with the same node as start and end point
[20] a tool (algorithm,executable) that calculates the behavior of a model (simulation)

what	to	when
Properties of the slave	Master	Initialization phase
Status of the slave	Master	After communication step
Input values & derivatives of the slave	Slave	Before communication step
Output values & derivatives of the slave	Master	After communication step & after initialization
Control commands (simulate communication step/finish simulation)	Slave	(simulation phase/closing phase)

Table 2: Master-slave information exchange in the different phases

A simple example of a master algorithm could be described as follow:

- Communication step size $hci = tc(i+1)$ - tci = hc $\forall$ i in R and tc: time communication
- For all the slaves, the first input value is chosen by the master $Utstart$ =0
- Increase i till time $tstop$ and do:
  - input values $Utci$ and the communication step size are transfered to slaves
  - slave simulation begin and output values $Ytci$ are transfered to the master
- Distribute the output values $Ytci$ by the master at each communication point tci to the slaves inputs $Utci$ according to a connexion graph for the communication steps $[tci\ ,\ tc(i+1)]$

The master-slave co-simulation has drawbacks. The exchanges (read, write) between the two simulators are done alternately. This synchronization mode does not allow to run the simulators in parallel. So when the master simulator is running, the slave simulator is stopped, and vice versa. (cf. Figure 24) gives an example of this procedure. This type of co-simulation is well suited for synchronized data exchanges while it is not very functional for applications based on the execution control.

2.2 bus-based distributed architecture

Distributed simulation processing must ensure that existing temporal causal[21] relationships are reproduced in the equivalent sequential execution.
These distributed simulations assume that simulation is considered to be composed of a set of Logical Processors (LPs) that communicate by exchanging dated messages (events).
We distinguish two types of synchronization:

[21]causality defines that the future can not influence the past. The state of the system at the moment (t) is independent of anything that can occur at any date (t') greater than (t)

Figure 24: absence of parallelism in master-slave

- **Pessimistic (or conservative) approach**: In this solution, the LPs are connected by links (or channels) that define the structure of the influence relationships between LPs. It is assumed that the LPs issue outgoing messages (events) in a nondecreasing order and that the network ensures that the messages are received in the same order as when transmitting. This assumption guarantees that the last message received on a link is a minimum date compared to the date of the next messages to be received on this link. Finally the LP selects the minimum date message among those received on these influencer links and its local messages. However, a problem occurs in the conservative synchronizations if one of the links is empty when determining the next message to be processed. In this case, we do not know the date of the next message to be received by this link and the LP waits indefinitely for this information, this results in a deadlock.
  A null message, as a solution for a deadlock, has no content other than a date. When a processor sends a dated message to one of its outputs, it also sends a Null message of the same date to its other outputs.

- **Optimistic approach**: In contrast to the previous approach, the optimistic approach allows violation of the causal constraint. The LP deals with events "to its knowledge", this local and therefore partial knowledge of the events to be treated may lead to the omission of certain external events and the non-respect of causality [Samadi, 1985].
  If a causal constraint is violated (reception of an event whose date is earlier than the current local date of the LP), the simulation "returns to its past". A Rollback mechanism is then triggered [Jefferson, 1985].
  The shortcomings of this optimistic approach come in particular from the execution time spent making rollbacks, which is often too important, and the need to memorize the events received, transmitted and the states reached.
  It should be noted that there is a third approach combining the first two depending on the cost of using one method or another on the simulation.

The bus-based distributed model overcomes the limitations of the master-slave model. The data exchange takes place on the call of an input/output (I/O) procedure by a simulator. Each simulator sends its data and stays on hold for receiving data from the other side. It relies on a co-simulation bus used as a communication protocol (cf. Figure 25).
The I/O procedures of each simulator take care of extracting its data and communicate them to

the co-simulation bus, as well as directly calling another simulator as a slave. So each simulator acts as master.

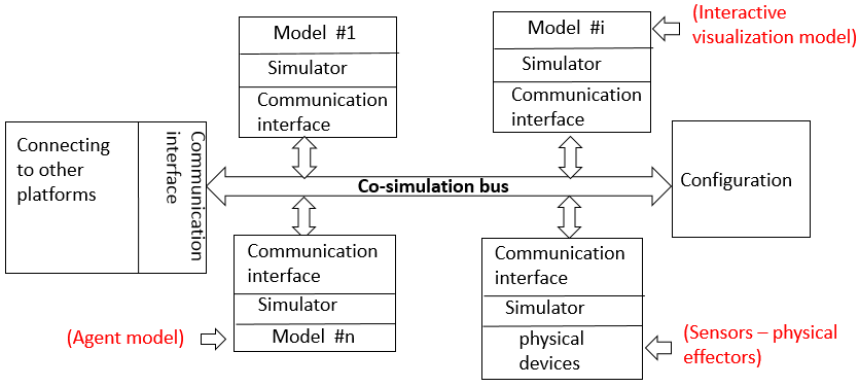

Figure 25: Example of a distributed co-simulation platform

In our case, the FMI standard used for the structural interoperability follows a master-slave architecture, and the master algorithms are not standardized with FMI. We studied some master algorithms developed in the MODELISAR [22] project e.g. by tool vendors. As soon as we realized that a generic master algorithm is a difficult task (impossible), and as we want to be flexible in the design of each component and allow a certain modularity of the global system, keeping the possibility to interchange or add new elements without questioning the development of each sub-system. and as we proposed to achieve a semantical interoperability by adding data mediation bricks to our open system. We went for a distributed architecture instead of developing a *specific* master algorithm, then our FMUs can be managed and synchronized under this approach.
In this way we can also offer a gain in performance distributing the calculation over several processors or several machines. In order to avoid the weakly coupled simulators paradigm without which this gain wouldn't be significant. We extended our distributed model in a way to optimize the parallelism through a way of storing the exchanged data. Thus, the simulators read and write at their convenience, throughout their execution, in the co-simulation bus and do not remain waiting for data. However, this synchronization mode introduces new constraints at the level of the communication protocol.
The complexity of this model focuses on the co-simulation platform: managing access to the co-simulation bus and coordinating the data by the bus controller. The co-simulation bus characteristics are described below.

2.3 Co-simulation bus

A bus, in the computer field, as opposed to a point-to-point link, is a communication system shared between different modules, whether hardware or software.
A multitude of communication bus implementations exist in the different digital systems (automobile, operating systems,etc.), some are working on the hardware component of the connection (cables, etc.) which is not our case. We limit our work on the software component that defines the

[22] `http://www.modelisar.org`

communication protocol.
We distinguich two roles provided by our co-simulation bus:

- link the different simulators
- organize the exchange of data between them

These roles are insured by two elements:

- **the communication medium:** It's the software element that will allow the transfer of data between the simulators. Several solutions exist to set up this communication, each with its advantages and its disadvantages. For example, a shared local memory has the advantage of being very efficient with respect to the transfer time but prohibits the distribution of simulators on several machines. For a network communication, the most widespread solution is the Socket , providing the ability for interprocess communication using file descriptors. This is a software interface with the network services of the operating system on which the data controller is connected.

- **the data controller:** coordinates the exchange of information between the different simulators of the environment. Depending on the synchronization model described in the subsection 2 used at the simulators level, the controller will have to route the data from the sender to the receiver, using a communication mechanism (socket for example). This requires the use of communication services at different levels. We distinguish:

  - High-level communication services, such as CORBA described in the subsection 2.1.1, have the advantage of simplifying the exchange of data between modules by offering an abstraction of the communication. Thus, a simulator can be deployed on a remote machine without regardless its implementation which remains intact. This abstraction however entails an additional cost of communication time since it relies on services of lower levels

  - Low-level communication, such as TCP or UDP used as transport protocol, provides end-to-end connectivity specifying how data should be formatted, addressed, transmitted, routed, and received at the destination. They are much faster, although much less flexible. The choice of the mode of communication will be done according to the needs in terms of performance and flexibility.

  No need for receivers address since the data is available to all simulators connected to the bus. In our case where several simulators are connected to the bus, their parallel execution let each of them initiate, at any time, an I/O procedure.

Co-simulation allows us to couple models of different origins, expressed in different formalisms. Thus, the transformation of existing simulators is limited to the definition of communication interfaces. Each simulator, as said before, is a black box mock-up of a constituent system, developed and provided by the team/researcher that is responsible for that system. This allows each team/-supplier to work on its part of the problem with its own tools, as they use to do in th neOCampus project, without having the coupled system in mind.

3 Co-simulation tools

There is a lot of work and different engineering domains that has been widely reported in the literature, as shown in (cf. Figure 26), in which the co-simulation has been applied. However, the unexplored potential is recognized in a number of completed and ongoing projects that address co-simulation MOELISAR[23], DESTECS[24], INTO-CPS[25], ACOSAR[26], ACoRTA[27].

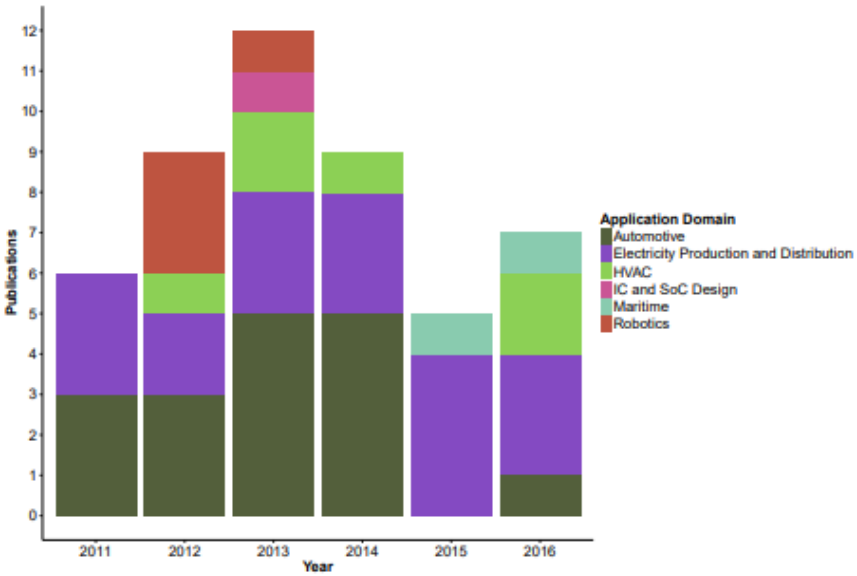

Figure 26: Research publications of co-simulation applications over the past five years. Source: [Gomes et al., 2017]

From the co-simulation tools list, we can cite:

[23] https://itea3.org/project/modelisar.html
[24] `http://www.destecs.org/`
[25] `http://into-cps.au.dk/`
[26] `https://itea3.org/project/acosar.html`
[27] http://www.v2c2.at/research/ee-software/projects/acorta/

- A coupling tool named DACCOSIM which is developped as a cross-platform version relying on JAVA and as a Windows version using C++ and QTronic SDK. It's able to perform Distributed simulations and multi-core simulations [Galtier et al., 2015]

- MDPCosim framework which is a master-slave co-simulation environment [Günther et al., 2012]

- SystemC/Matlab co-simulation tool for networked control system is decribed in [Quaglia et al., 2012]

- A comprehensive co-simulation platform for cyber-physical systems used in [Al-Hammouri, 2012] for the integration of two tools: Modelica, and NS-2

- A coupling tool named NCSWT describes in [Eyisi et al., 2012] the coupling of two tools: Matlab and NS-2 using HLA standard

- A coupling tool named RoboNetSim as an integrated framework for multi-robot and network, used in [Kudelski et al., 2013] to integrate three simulators (ARGoS, NS-2 and NS-3

Two other co-simulation platforms were studied and used in our work, we separate them according to their fields of use as follow

3.1 Business field

The CosiMate[28] environment offers a complete simulation environment dynamically linking heterogeneous simulators and which can be extended to different simulation environments on different platforms. It is an open architecture based on a co-simulation bus as described in the subsection 2.3. It provides synchronization methods that take into account the different behaviors of the languages and the simulators used. Thus, when a simulation is performed on a network, CosiMate considers the intrinsic constraints of the communication medium. CosiMate adapts to the network configuration, offering a co-simulation based on a multi-client multi-server to avoid unnecessary communications between simulators instantiating local routers for each computer in the co-simulation. And when different parts of the system are co-simulated at different levels of abstraction, it is necessary to add adapters (wrappers) to the compatibility of data exchange between models at different levels. CosiMate meets our needs by offering two co-simulation working modes:

- **Event driven mode:** The router (that manages the data exchange and synchronizes the simulator) does not deal with any notion of time. This mode of communication makes it

[28] `https://site.cosimate.com/`

possible to establish a connection between the event simulators (HDL simulators, UML models) and sequential simulators (code C for example). The data is transmitted once available on the CosiMate bus. Then the router validate the transmission between the sender and the receiver (without checking if this last one read the data). CosiMate is flexible enough to support different communication protocols.The data is transmitted once available on the CosiMate bus, the valid transmission of the router between the sender and receiver (does not check if the recipient has read the data). CosiMate is flexible enough to support different communication protocols. We distinguich:

- locked input ports: until an event is sent from a second simulator over the cosimate bus
- unlocked input ports: events are propagated (the receiver is configured to read the data/events at the right time; maybe with signal coming from the cosimate bus)
- use buffered and unbuffered variables: for the first case a FIFO[29] (First in, first out) is instantiated and its depth defined

- **Synchronized co-simulation mode:** The router synchronizes the models (subsystems to simulate in a simulation environment that provides CosiMate input / output ports) taking the minimum time. This mode is suitable for simulation engines using solvers (such as Matlab / Simulink). If the simulators are synchronized (T1 = T2 = ... = Tn), CosiMate is synchronized to a single frequency during co-simulation of the bus. If the simulators are out of sync, the simulators are interconnected at a single frequency (Min (T1, T2 .., Tn)), or are interconnected in groups (each group uses a single frequency to exchange data in the co-simulation bus)

3.2 Research field

MECSYCO[30] (Multi-Agent Environment for System Complex CO-simulation) is a software for simulation and modeling of complex systems. It allows numerical simulations from existing and heterogeneous simulators. It relies on the universality of the DEVS formalism in order to integrate models written in different formalism. This integration is based on a wrapping strategy in order to make models implemented in different simulation software inter-operable. The middleware performs the co-simulation in a parallel, decentralized and distributable fashion thanks to its modular multi-agent architecture.

[Vaubourg et al., 2015] shows how the Agents and Artifacts approach used by MECSYCO makes it possible to simultaneously manage these different challenges through a concrete application case of smart grids.
MECSYCO adopts a decentralized coordination where each simulator is associated with a local scheduler. This decentralized coordination is two-fold:

[29] the first element added to the queue will be the first one to be removed
[30] http://mecsyco.com/fr/

- A first one specifies what and when to exchange between models. This corresponds to the description of the information to read or to write from a model (what), and at which time of the local simulation it has to be done (when). This part is assumed by a symbol used in MECSYCO called the M-Agent whose behavior corresponds to the Chandy-Misra-Bryant algorithm which checks for deadlock in a distributed system. This behavior enables the global simulation of the multi-model in a decentralized way. This part is generic

- The second one describes how to execute one step of the local simulation. It links inputs and outputs of the M-agent with the concrete representation in the simulator and asks the simulator to execute one step of simulation

This platform uses a multi-agent architecture, which will be described in the next chapter, to achieve the rigorous integration of the various components of the simulation.

After describing in the first chapter the context of work which is the smart campus and defining it as a complex system with all the issues this categorization embodies, we explained how the collaboration is hard to realize. We then introduced the concept of interoperability between campus business tools for dynamic simulation. We detailed the different interoperability solutions (data, models and computation codes), and our interoperability approach (two levels: Structural and Semantic) was described and justified. For that, in the structural level, a standards interfaces table was constructed and the choice was made on FMI (Functionnal mockup interface) which will be adapted to our needs. We introduced the ways we approached the semantical interoperability (2 ways). The first way, data mediation, was described and will be more detailed in the 3rd chapter. The second way, Dynamic mediation using artificial intelligences, was left to the 3rd chapter as it needs the introduction of multi-agent systems, and the description of how and why we used them.

4 Classification of AI approaches

In order to justify our approach which will is described in the next chapter. We first needed to classify artificial intelligence technologies based on the AI knowledge Map (AIKM) developped by "Francesco Corea" see (cf. Figure 27), then make a choice of the one that will be used (Adaptive multi-agent systems AMAS described in 2).

Due to the complexity of our work which is irregular and evolve in time, the use of an artificial intelligence was made as an alternative solution of creating a single ontology 3.2.1 containing representations of every term used in every subsystem.

4.1 AI knowledge Map (AIKM)

In the figure 27 two axes were distinguished:

- the AI Paradigms[31] setting apart:

[31] approaches used by AI researchers to solve specific AI-related problems

- Logic-based tools: tools that are used for knowledge representation and problem-solving
- Knowledge-based tools: tools based on ontologies and huge databases of notions, information, and rules
- Probabilistic methods: tools that allow agents to act in incomplete information scenarios
- Machine learning: tools that allow computers to learn from data
- Embodied intelligence: engineering toolbox, which assumes that a body (or at least a partial set of functions such as movement, perception, interaction, and visualization) is required for higher intelligence
- Search and optimization: tools that allow intelligent search with many possible solutions

- the AI Problem Domains[32] setting apart:
  - Reasoning: the capability to solve problems
  - Knowledge: the ability to represent and understand the world
  - Planning: the capability of setting and achieving goals
  - Communication: the ability to understand language and communicate
  - Perception: the ability to transform raw sensorial inputs (e.g., images, sounds, etc.) into usable information

We focus in our work on Distributed Artificial Intelligence (DAI[33]) which as we can see can be used in all the AI problem domains. More precisely we focus in this "DAI subset" on Multi-agent systems (MAS) where collective behaviors emerge from the interaction of decentralized self-organized agents, and which will be adapted to respond to our need, for instance it can also include machine learning (an agent could be a machine learning algorithm).

[32] type of problems AI can solve

[33] class of technologies that solve problems by distributing them to autonomous “agents” that interact with each other

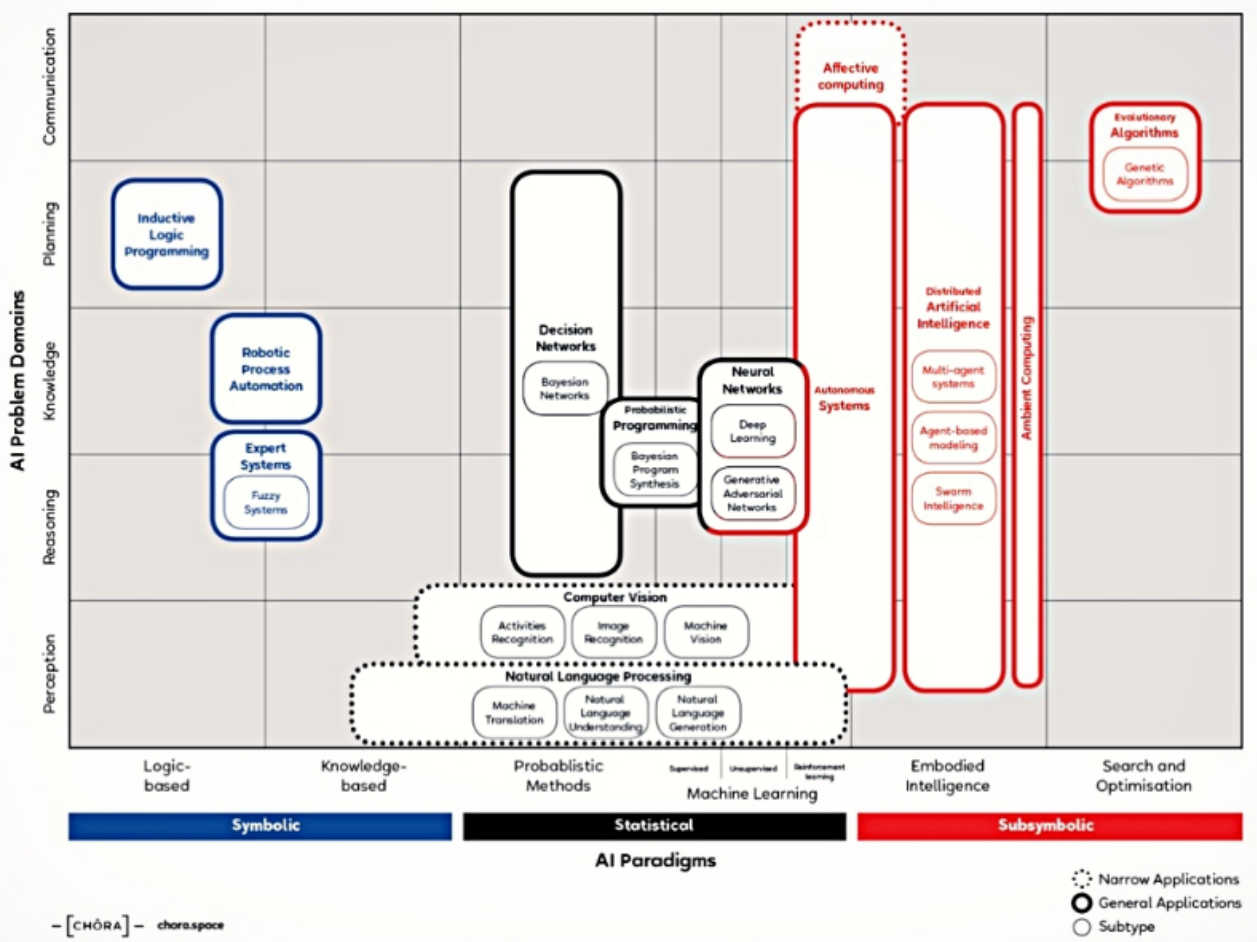

Figure 27: Artificial Intelligence Knowledge Map, source: `https://cognitiveworld.com/article/ai-knowledge-map`

4.2 Adaptive Multi-Agent Systems (AMAS) and Other semantic approaches

We assume that Ontology, as the most used semantic approach, is more expressive, formal, structured, but we should mention that it needs a domain expert knowledge and that the update cost is high compared to the AMAS. Which are less expressive but don't need any expert knowledge which is one of the strong point that leads us to use them in addition to the fact that they are autonomous and adaptable see table 3.

Ontology	AMAS
+ Expressive	- Less expressive
+ formal & structured	- Less structured (but known algorithm)
- Expert Knowledge (Domain)	+ Agnostic (no need for expert knowledge)
- Update cost (High)	+ Autonomous & Adaptive (Update)

Table 3: Ontology and AMAS pros (+) and cons (-)

In our point of view, with an ontology one understands exactly how learning is done how the results are produced in an explicit way comprehensible by the human, while with AMAS it is less structured, but the ontological apect is included but not in an expressive way (i.e. the meaning is rather made for the system).
The similarity of AMAS with the world of complex systems (AMAS is an example that has been proven) strengthens our choice. Despite the fact that AMAS doesn't give perfect results, but rather good, they work in any case, whatever the possible evolution of the system is, AMAS will always be able to respond.
In a subjective way, if we want to make a scale of expressivity where ontology will have 100 percent (white-box), and Neural network 0 percent (black-box), we can say that our AMAS are 50 percent expressive (Gray-box), due to the use of DREAM see 2.6, which we consider 70 percent expressive and AMOEBA 30 percent (because of the high number of contexts and dimensions) see 2.7.

As we chose the components approach for interoperability. We defined in this chapter the co-simulation as the coupling of several simulation units. We studied the synchronization models and defined the co-simulation bus, and compared the master-slave and distributed architectures then assert our choice. We then cited some co-simulation tools and detailed the most interesting for us.

In the next section we will introduce the steps to build our co-simulation framework and describe the artificial intelligence (AI) used to achieve the semantical interoperability level.

CHAPTER IV: Co-simulation framework

1 Co-simulation framework interoperability

A framework provides us with the certainty that we are developing an application that is in full compliance with the business rules, that is structured, and that is both maintainable and upgradable.

It allows us to save time by re-using generic modules in order to focus on other areas. Without, however, ever being tied to the framework itself.

Our approach for the design of the co-simulation framework interoperability is based on:

- a **co-simulation**: as the approach for the joint simulation of models developed with different tools (tool coupling). As described in 1 each tool treats one part of a modular coupled problem. The data exchange between these tools during simulation is restricted to discrete communication points. The subsystems are solved independently between these communication points.

- a **software components** approach which is defined by [Szyperski, 1996] as a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component is a software element that conforms to a component model. It's similar in our case to FMUs described in 3.1.5 which can be independently deployed and composed without modification according to a composition standard (FMI). This lead us to define A software component infrastructure as a set of interacting software components designed to ensure that a software system or subsystem constructed using those components and interfaces will satisfy clearly defined performance specifications.

- a **Structural interoperability** using the standardized interface FMI (Functional Mock-up interface) as described in 3.1.5. It's an interface standard for the solution of time dependent coupled systems consisting of subsystems that are continuous in time or time-discrete. It provides interfaces between master and slaves and addresses both data exchange and algorithmic issues.

- a **Semantic interoperability** using mediation for adaptation of the data which will be detailed in the two next subsection, first with the use of a mediation model as introduced in 3.2.1, then with the use of the adaptive multi-agent systems which will be detailed.

1.1 Interoperability based mediation model

As described previously the syntactic interoperability based on FMI standard is a prerequisite for semantic interoperability. Syntactic interoperability refers to the packaging and transmission mechanisms for data.
It's common for standards addressing simulation interoperability to be almost exclusively data centric. As in the protocol data units defined under IEEE Standard 1278 Distributed Interactive Simulation, and the Object Model Template of the IEEE Standard 1516 High Level Architecture. They define information exchange elements to be distributed among participating simulations.
Time management and data distribution management add some complexity; however, the basic interoperation function to make sure that information is distributed in a standard form and reaches the participating simulations at a given logical, simulation internal time. FMI for co-simulation used in our work consists of two parts:

- Co-Simulation Interface: a set of C functions for controlling the slaves and for data exchange of input and output values as well as status information.

- Co-Simulation Description Schema: defines the structure and content of an XML file, containing static information about the model (input and output variables, parameters, etc.) and the solver/simulator (capabilities, etc.).

The XML which is the current internet standard for document markup uses data delimiter (""). It conveys no meaning to the data other than to structure the data. Without a data dictionary to translate the contents of the delimiters, the data remains meaningless. While there are many attempts at creating data dictionaries and information models to associate with these data packaging mechanisms, none have been practical to implement, leading to the inability to exchange of data with meaning. XML helps however to structure any data so that the manner of processing the information will be interpretable from the structure.
It also allows detection of syntactic errors, thus allowing receiving systems to request resending of any message that appears to be garbled or incomplete. The information represented in one syntax may in some cases be accurately translated into a different syntax. Where accurate translation of syntaxes is possible, systems using different syntaxes may also interoperate accurately. In some cases the ability to accurately translate information among systems using different syntaxes may be limited to one direction, when the formalisms used have different levels of expressivity (ability to express information).

Semantic interoperability defined as the ability of computer systems to exchange data with unambiguous, shared meaning is then a requirement to enable machine computable logic, inferencing, knowledge discovery, and data federation between information systems.
This can be accomplished by adding data about the data (metadata), linking each data element to a controlled, shared vocabulary. The meaning of the data is transmitted with the data itself, in one self-describing "information package" that is independent of any information system. In our complex system we considered it impossible to create a single ontology containing representations of every term used in every subsystem. As first alternative we used a mediation model that describes,

see 3.2.1, the information shared by taking their semantics into account in the form of contexts of use.

The method we use to cope with this challenge of creating a common understanding of the information space is data engineering. Which supports a holistic view of data related efforts. Its concepts are based on the principles of federated and distributed databases. Focusing on the combination of the often independent and unaligned following disciplines:

- Data Administration which is the process of managing the information exchange needs that exist between subsystems, including documentation of the source, the format, context of validity, and fidelity and credibility of the data [Pettersson, 2004].

- Data Management which consists of defining and using rules, methods, tools and respective resources in order to identify, clarify, define and standardize the meaning of data through their relations.

- Data Alignment which ensures the existence of the data to be exchanged in the participating subsystems as an information entity or that the necessary information can be derived from the data available, using the means of aggregation or disaggregation.

- Data Transformation is the technical process of aggregation and/or disaggregation of the information entities from the component systems to match the information exchange requirements, including the adjustment of the data formats as needed [Tolk and Pullen, 2005].

As our components are implementing the FMI standard (e.g., Functional Mock-up Unit (FMU)). Each consists of one zip file containing:

- the XML description file

- the implementation in source or binary form (dynamic library)

This use of XML model description as common syntactical standard facilitates each of translation. The mediation process in essence becomes a translation among different dialects of XML. The data administration also can be directly supported thanks to the use of cosimate 3.1. This first contribution within our complex system using a data mediation model is described in more details below explaining the use of cosimate platform and presenting our case study for neOCampus project that shows how the framework helps to build the semantic interoperability of a cyberphysical system.

1.2 The co-simulation framework based component

1.2.1 Simulator

We chose the CosiMate environment which offers a complete simulation environment dynamically linking heterogeneous simulators and which can be extended to different simulation environments on different platforms. CosiMate provides synchronization methods that take into account the different behaviors of the languages and the simulators used. Thus, when a simulation is performed on a network, CosiMate considers the intrinsic constraints of the communication medium CosiMate adapts to the network configuration, offering a co-simulation based on a multi-client multi-server to avoid unnecessary communications between simulators instantiating local routers for each computer in the co-simulation. And when different parts of the system are co-simulated at different levels of abstraction, it is necessary to add adapters (wrappers) to the compatibility of data exchange between models at different levels. CosiMate meets our needs by offering two co-simulation working modes:

- **Event mode:** The router (that manages the data exchange and synchronizes the simulators) does not deal with any notion of time. This mode of communication makes possible to establish a connection between the event simulators (HDL simulators, UML models) and sequential simulators (code C for example). The data is transmitted once available on the CosiMate bus. CosiMate is exible enough to support different communication protocols.The data is transmitted once available on the CosiMate bus, the valid transmission of the router between the sender and receiver (does not check if the recipient has read the data). CosiMate is flexible enough to support different communication protocols.
- **Synchronized mode:** The router synchronizes the models taking the minimum time. This mode is suitable for simulation engines using solvers (such as Matlab / Simulink).

1.2.2 Cosimate-FMI

Cosimate allows the co-simulation between FMI and also non-FMI models. There are simulators which are not supported by Cosimate, thus the need to wrap them as an FMU component in order to plug them to our cosimate bus

1.2.3 System

The co-simulation of a complex system can thus be based on the joint simulation of all its subsystems. It also makes it possible to simulate the whole system by coordinating and exchanging data calculated and interpreted by each subsystem, in order to obtain a result which does not modify the functionality of the implementation of the future system. Among our different simulated models, we distinguish:

- Functional simulation allowing the validation of the aspects of the system which are independent of time, and here we can dissociate the sequential simulators and the event simulators.
- Temporal simulation which aims to exchange data in time windows. Knowing that if data is not consumed in a given time window, it may be lost. A synchronization model is therefore necessary to coordinate the parallel execution of our different simulators.

1.3 neOCampus case study

1.3.1 Different simulators used

As described in the subsecion 1.3 there are several simulators and sensors scattered around the campus

We therefore have: In one hand several simulators on a different fields using different kind of data.
One is working with **Matlab Simulink** on the energetic consumption using a black box neural network Heat pump model to ensure a comfortable desired in the rooms by heating and cooling when it is necessary. He is interacting with the outdoors getting from sensors the Electric power (Kw) and the temperature coming from the building and generating with a specific time step.
The second simulator is working with **powersims** toolbox of Simulink [Khader et al., 2011] using Maximum Power Point Tracking making it possible to follow the maximum power point of a non-linear electric generator. He is interacting with the outdoors getting the values of the Photovoltaic current and the voltage and generates Converter control setpoint.
The third simulator using **Contiki** [Dunkels et al., 2004] which is an operating system for networked, memory-constrained systems with a focus on low-power wireless Internet of things devices using Cooja which allows large and small networks of Contiki motes to be simulated in order to evaluate the performances (energy, delays) of IOT networks using the protocol CCN (content centric Networking) applied on a network of sensors. Developed in C++ under linux OS. The way it interacts with the outdoors is using interest (requests sent by users containing the name of the data such as the temperature) and generating the value of this data.
In the other hand the **collection of sensors data** is stored in a NoSQL **database** (mongodb).

1.3.2 Co-simulation engine

The design approach of neOCampus is necessarily scalable and adaptive, which directs our work towards the development of global and open simulation environment. As we said before we adopted the component approach and described the general FMI's way of working. This last follows a master slave architecture, and we mentioned that a master algorithm needs to be defined in order to synchronize the simulation of all subsystems and to proceed in communication steps. That the data exchange between subsystems is connected via MPI (Message Passing Interface), TCP/IP (Transmission Control Protocol/Internet Protocol), Sockets, and that the mapping between outputs to inputs has to be initialized. Cosimate, as described, makes us save the efforts of dealing with synchronization between our subsystems. To perform this integration, CosiMate provides libraries to make the customization easier. The libraries contain (1) I/O ports compiled and described for the simulator/language used. (2) I/O ports description. This description depends on the environment in which the ports are to be used: for example, a header file for C/C++ language is provided. In our case and according to the different simulators mentioned previously, we constructed an FMU's component for each one either by using FMI toolbox like for MATLAB/Simulink or PSIM, or a wrapper using FMI Library from Modelon for the simulator using Contiki. We made it easy to connect all the simulators knowing that for example, Simulink and PSIM are supported by Cosimate but Contiki is not. But the CosiMate FMI connector can load and run all FMI models compatible with FMI 1.0/2.0 for the Co-simulation mode. As we said before each of our FMU files is a zip file that contains a file named modelDescription.xml and one or more platform-dependent shared libraries. The XML files are used to describe how a model running in a simulation environment is

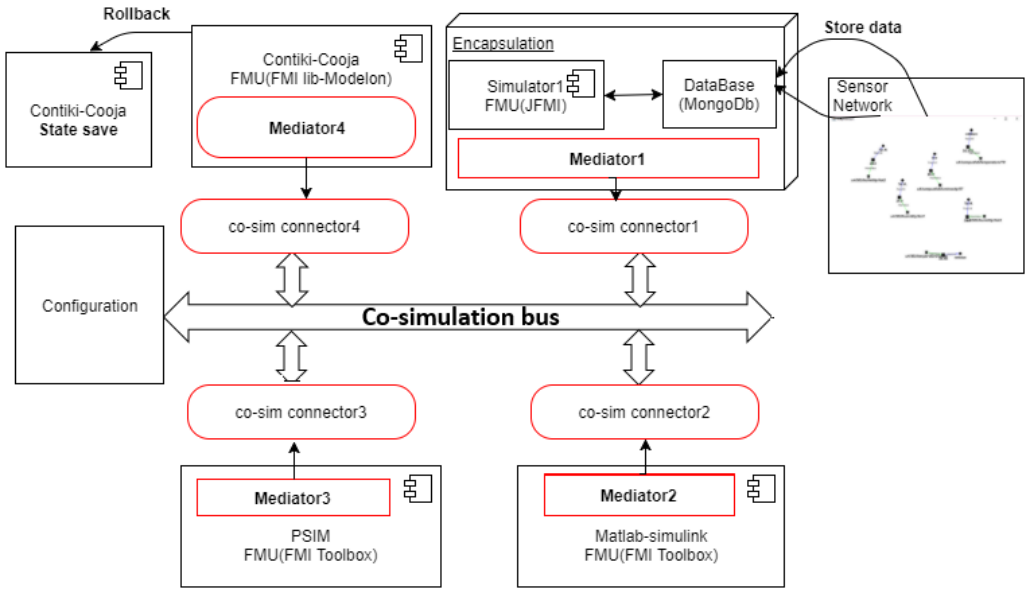

Figure 28: components mediation

connected to the CosiMate bus. We should mention that CosiMate allows execution in the native simulation environment, users can easily work in their familiar environment controlling, debugging, and monitoring simulations as if they are running in a stand alone mode integration. We can also use remote procedure: if the model is to be run on a remote machine. The CosiMate Spy tool is used as an observer of the co-simulation components and processes. It acts as a reader of the CosiMate bus without modifying data exchanges or simulations synchronization during the co-simulation.

1.3.3 Mediator components

One of the problems encountered is the mediation part, since we want to achieve a semantic interoperability we offered the possibility for each simulator to decide of the way it wants to receive information and depending of the components it's talking to (if it already knows them) to convert its output. For that we encapsulate a mediator with each component before connecting it to the cosimate bus. We added some procedures which allow us to copy and later restore the complete state of an FMU component providing a mechanism for rollback (inspired from the optional functions of the API of FMI 2.0). For our sensor network and as we said that it uses a database to store raw data. This has led us to develop a java simulator (using Mongodb Java Driver) that bridges between the mangodb database and our cosimate bus. We encapsulate the database and our simulator using JFMI (a java wrapper for FMI). So this virtual encapsulation offers capabilities of data mediation and distributes query processing. So the other simulators have no need to know about the database type and location and data can be accessed easily see (cf. Figure 28).

We have implemented our architecture and our modeling works well, we took as example the 4 simulators including Contiki which is not compatible with Cosimate and for which we had to generate a slave FMU. Our database was encapsulated using JFMI. We were able to solve not only

these structural problems but we added mediators to our platform in order to achieve semantic interoperability. It is necessary to mention that our framework allows the integration of all types of simulators and that for the non FMI and even if they are not supported by cosimate the use of a wrapper is enough to envelop them with a c code in order to connect them to the cosimate bus.

In this first work [Motie et al., 2017] we designed a co-simulation framework interoperability performing a **co-simulation** based on a black box **component approach**. We defined communication ports and guaranteed the possibility of connection by respecting data types and the direction of the ports following the **FMI standard**. We designed a **mediation approach** to ensure the unambiguous information exchange. We noticed that the semantic interoperability based on mediation was integrated to our framework in an ad-hoc way. This took a heavy work investment as each component needed a hand-made encapsulated mediator.

we propose a *dynamic* mediation for adaptation of the data which allows to go further towards the development of global and open simulation environment. The idea is to take advantage of the self-adaptation and openness capability of Adaptive Multi-Agent Systems (AMAS), and to automate the process using *DreAMoeba*, the extension of DREAM[34] [Belghache et al., 2017] with AMOEBA[35] [Verstaevel et al., 2017], as a black box component in the co-simulation framework, which is able to link dynamically correlated inputs and outputs and hence to continuously adapt the structural interoperability and carry out the semantical one.
Before presenting the two collective artificial intelligence based tools DREAM and AMOEBA, we may describe our motivations to use adaptive multi-agents systems. Which is inspired from the analysis of the definition stated by the New England Complex Systems Institute (NECSI [36]): "Complex Systems is a new field of science studying how parts of a system give rise to the collective behaviors of the system, and how the system interacts with its environment. Social systems formed (in part) out of people, the brain formed out of neurons, molecules formed out of atoms,the weather formed out of air ows are all examples of complex systems. The field of complex systems cuts across all traditional disciplines of science, as well as engineering, management, and medicine. It focuses on certain questions about parts, wholes and relationships. These questions are relevant to all traditional fields.".

To overcome these difficulties, the system has to be self-adaptive. As designers of complex systems, we have been taking inspiration from natural systems in which complex structures or behaviours emerge at the global level of a system from interactions among its lower-level components. Since a Multi-Agent System is defined as a macro-system composed of autonomous agents which pursue individual objectives and which interact in a common environment to solve a common task, it can be viewed as a paradigm to design complex applications.

Based on the data engineering method described in the subsection 1.1, the combination of the data administration, data management, data alignment and data transformation disciplines, and the use of a common syntactical standard, in order to achieve the semantic interoperability level. We showed that the main intellectual process in data engineering is the data management process,

[34] *DREAM* stands for *D*ynamic data *R*elation *E*xtraction using *A*daptive *M*ulti-agent system
[35] *"AMOEBA"* stands for *A*gnostic *MO*dEl *B*uilder by self-*A*daptation
[36] http://www.necsi.edu/guide/study.html

in which data elements are identified and described, and equivalent expressions of information are mapped to each other.

[Tolk, 2004] described a model based data management (MBDM), and [Tolk, 2004] used it in an XML mediation services. In [Tolk and Pullen, 2005] the MBDM uses a distinguished common reference data model as the hub for mediation, where the reference data model to which all data models are translated is enhanced and extended to define standardized data elements in case of need. The following cases are distinguished:

- no action is needed beyond using this mapping as the unambiguous definition for the semantic interpretation of the information element, if this last one can be described by a standardized data element already in the common reference data model
- add new relations among the defining standard data elements in order to generate the aggregation, if the information element is described by more than one standardized data element
- a new piece of information needs to be modeled with the common standard (extension of the standard), if no conterpart exist in the common reference model for the information element
- increase the resolution of the standardized data element if several information elements can be grouped together and mapped to one standardized data element

There is also an increasing interest in Multi-agents systems (MAS) technologies and their applications on data analytics [Cao et al., 2013]. Several try to use concepts like swarm intelligence, self-organising maps, etc. However, all these new techniques, including our mediation model 1.1, are still domain dependent and do not handle changes in the data. We aim to tackle this by applying the AMAS technology and its mechanism of adaptation through cooperation.

1.4 Managing Data in Dynamic Complex Systems

Since we deal with the packaging transmission of the data exchange between the heterogeneous components in the structural interoperability level, and the meaning, the variety (structured data which held unchallenged hegemony in analytics, unstructured data like text and human language, and semi structured data like XML, RSS feeds), and the velocity (rapid generation of data that arrives in batch as for the sensors in the campus) in the semantic interoperability level. The complexity, due to data dimension and variety, is a big issue, limiting the use of the conventional data analytics processes which don't allow the modification of already loaded data on the fly.

A simple way to model this data processing pipeline used for data coming from data generative devices is described in (cf. Figure 29) where we distinguish:

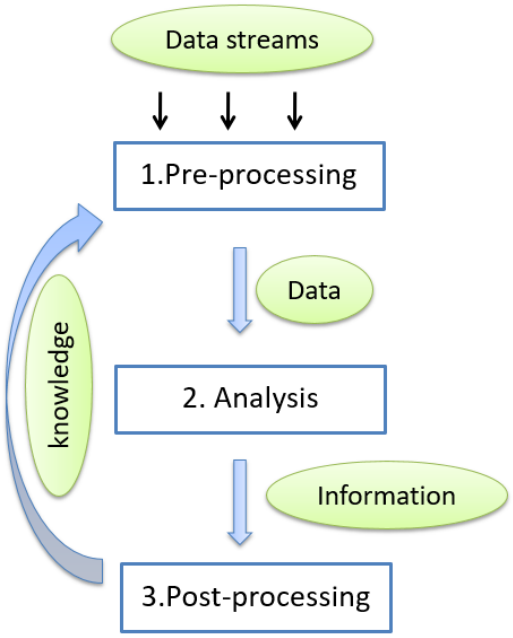

Figure 29: Conventional data analytics pipeline

- **Input / pre-processing** which is the transformation step of the incoming raw data into data by putting them in a nice ready-to-process form by gathering (data files, databases, data streams), integrating (combine the different sources), cleaning (noise, redundancy, inconsistency, etc.), reducing (Dimension Reduction, Feature Selection, Instance Selection, Discretization) the raw data

- **Analysis** also known as data mining, which is the center of the process. Thanks to a plethora of mining algorithms [Han et al., 2011], one can extract (explore and find) relevant information (clusters, association rules, regressions, etc.) from the data

- **Output / post-processing** which is the last step of the processing pipeline, in which the user produces his own knowledge about the incoming data by interpreting or annotating the extracted information by means of an intuitive visualization after that information has been evaluated and selected.

The distributed pipeline see (cf. Figure 30), was proposed to bypass this rigidity through processing time reduction by means of parallelism; for example with the help of the MapReduce pattern and its famous Hadoop framework [Dean and Ghemawat, 2008]. However, the main issue about dynamics still remains.

The AMAS technology, described in the section 2, with the cooperative interaction process of its autonomous agents,gives us the means to break down this rigidity and decentralize the data

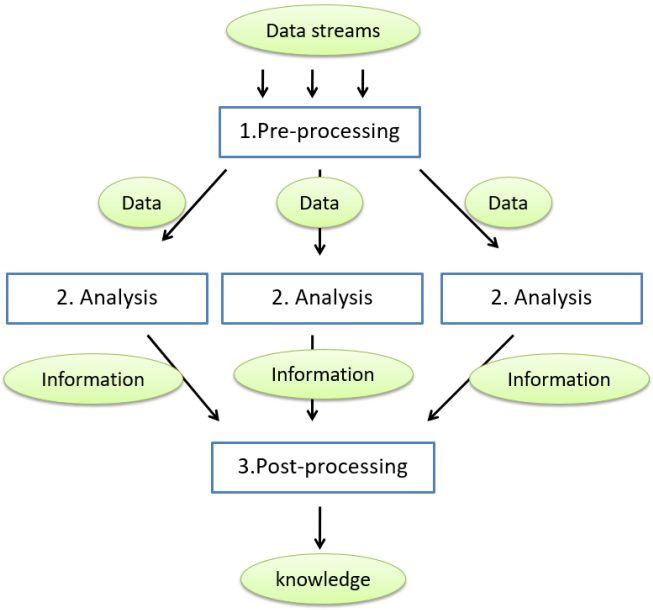

Figure 30: Distributed data analytics pipeline

analytics process see (cf. Figure 31). This results in data analytics tasks interaction, mainly through communication,and then each task can help and work together with other tasks for the sake of the continuous real-time adaptation of the analytic process to data changes.

2 Collective Artificial Intelligence for Semantic Interoperability

In this section, we present the tools used to achieve our new semantic interoperability and the theory behind them.

2.1 Multi-Agent Systems Theory

A multi-agent system (MAS) [Wei et al., 1999] is a distributed system composed of several autonomous software entities (the agents), interacting among each others (usually by sending information and request messages) and with their environment (by observing and modifying it). The autonomy of an agent is the fundamental characteristic that differentiates it from, for example, the computer science concept of object. While an object is a passive entity encapsulating some data and functions, waiting to be solicited, an agent is capable of reacting to its environment and displaying pro-activity (activity originating from its own decision). From this comparison it should be clear that the concept of agent is, like the concept of object, the building brick of a paradigm which can be used to model a complex reality in a bottom-up way, relying only on a limited and localized knowledge of the environment for each agent. And indeed, agents have been used in a great variety of fields, a fact which can contribute to explain the difficulty to produce a unified definition of the

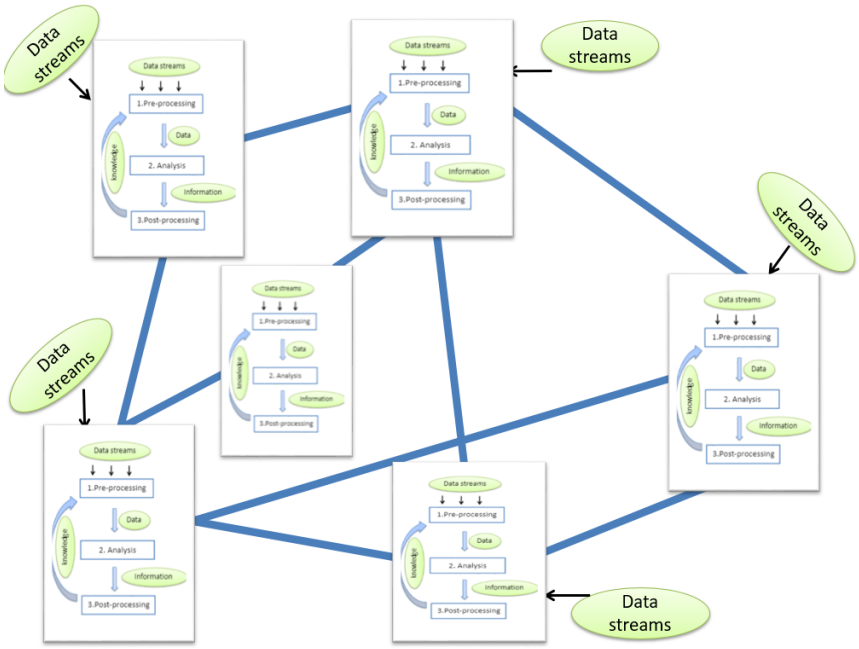

Figure 31: AMAS based data analytics pipeline

concept.

While it is not true for all MAS, some interesting properties can be achieved when taking advantage of the autonomy of the agents. This autonomy, coupled with an adequate behavior of the agents, can lead to systems able to adjust, organize, react to changes, etc. without the need for an external authority to guide them. These properties are gathered under the term self-* capabilities [Serugendo et al., 2011] (self-tuning, self-organizing, self-healing, self-evolving...). Not all MAS necessarily present all of these self* capabilities but, as a result of building a system from autonomous and locally situated agents,many MAS will exhibit them to some degree. Consequently,MAS are often relevant for dynamically taking into account changes in their environment. For example, a MAS in charge of regulating the traffic of packets in a computer network could be able to react efficiently to the disappearance of some of the relay nodes.

MAS have been applied to a great variety of fields: social simulation, biological modeling, systems control, robotics, etc. and agent-oriented modeling can be seen as a programming paradigm in general, facilitating the representation of a problem.

2.2 Adaptive Multi-Agent Systems

A particular approach to MAS relying strongly on self-* [37] properties is the AMAS technology and underlying theory [[Georgé et al., 2004]; [Gleizes et al., 1999]]. A designer following this approach

[37] self-tuning, self-organizing, self-healing, self-evolving...

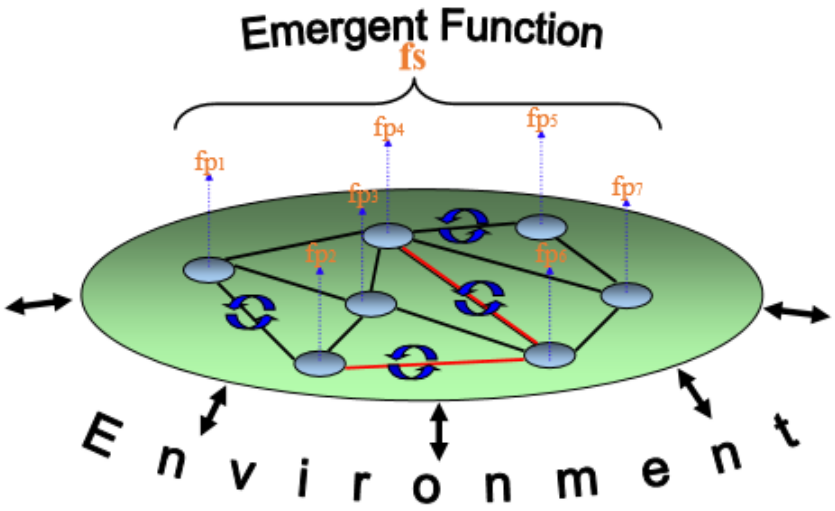

Figure 32: Adaptation: changing the function of the system by changing its organization

focuses on giving the agent a local view of its environment, means to detect problematic situations and guidelines to act in a cooperative way, meaning that the agents will try to achieve their goals while respecting and helping the other agents around them as best as they can. The fact that the agents do not follow a global directive towards the solving of the problem but collectively build this solving, produces an emergent problem solving process that explores the search space of the problem in original ways.

2.3 Adapt the System by its Parts

In our approach, we consider that each part P_i of a system S achieves a partial function f_{Pi} of the global function f_s see (cf. Figure 32). f_s is the result of the combination of the partial functions f_{Pi}, noted by the operator "o". The combination being determined by the current organization of the parts, we can deduce $f_s = f_{P1} \circ f_{P2} \circ ... \circ f_{Pn}$. As generally $f_{P1} \circ f_{P2} \neq f_{P2} \circ f_{P1}$, by transforming the organization, the combination of the partial functions is changed and therefore the global function f_s changes. So, enabling a MAS to self-organize consists in enabling the agent to change inside the organization. The global function realized is the result of the organization between agents in the system. This reorganization technique can be extended with two other related techniques: self-tuning (parts can modify the parameters defining their behaviour) and self-evolution (parts can appear and disappear when needed). To ensure that the system will generate emergent behaviours, according to the definition of emergence and to be able to control this emergence, it is necessary to provide to the agents a local criterion which enables them to self-organize. This requires both a theoretical and engineering framework.

Cooperation is the engine of the self-organization processes taking place in the system and the heart of our bottom-up method. Cooperation is classically defined by the fact that two agents work together if they need to share resources or competences [Ferber, 1999]. We add to this definition,

the fact that an agent locally tries on one hand, to anticipate problems and on the other hand to detect cooperation failures called Non Cooperative Situations (NCS)[38] and try to repair these NCS [Picard et al., 2005]. To anticipate NCSs, the agent always chooses the actions which disturb other agents it knows the less. In other words, the agents, by trying to always have a cooperative attitude, act by reorganizing their acquaintances and interactions with the others agents.

The objective is to design systems that do the best they can when they encounter difficulties. The designer has to describe not only what an agent has to do in order to achieve its goal but also which locally detected situations must be avoided and when they are detected how to suppress them.

2.4 Behaviour of an AMAS Agent

A cooperative agent in the AMAS theory has the four following characteristics. First, an agent is autonomous. Secondly, an agent is unaware of the global function of the system; this global function emerges (from the agent level towards the multi-agent level). Thirdly, an agent can detect NCSs and acts to return in a cooperative state. And finally,a cooperative agent is not altruistic (it does not always seeks to help the other agents), but benevolent (it seeks to achieve its goal while being cooperative).

Agents have to be able to detect when they are in an NCS and how they can act to come back in a cooperative situation. Agents also always try to stay in a cooperative situation and so the whole system converges to a cooperative state within and with its environment.

The main information an AMAS agent uses for its decision process is a specific measure called *criticality*. This measure represents the state of dissatisfaction or urgency of the agent regarding its local goal. Each agent is in charge of estimating its own criticality and providing it to the other agents[39]. The role of this measure is to aggregate into a single comparable value all the relevant indicators regarding the state of the agent. Having a single indicator of the state of the agent simplifies the reasoning of the agents. In addition, this mechanism has the interesting property of limiting the information transmitted to the others agents, which can be of interest in case of a large distributed systems where data privacy, data volume and computational complexity are issues.

With this additional information, each agent can and has to choose to cooperate with the most critical agent he is aware of. This leads to a very powerful heuristic to cut through a search space so as to drive the system to the expected state, effectively achieving a decentralized process that can be qualified as *emergent collective problem solving*.

This describes the typical decision process of a generic AMAS agent. But the NCS and the actions which could be applied to solve them are not generic: designers have to write their own specific NCS set and related actions for each kind of agent they wish the system to contain. Moreover,

[38] when a perceived signal is not understood or is ambiguous, perceived information does not produce any new decision, the consequences of its actions are not useful to others

[39] In open and untrusted environments, there exists several mechanisms to tackle uncertainty on exchanged information. This is often the case in System of Systems approaches. Inside a given system where each agent has been designed for the same stakeholder, each agent is assumed to provide the most trustful and accurate information

designers have the task to provide the agents which adequate means to calculate their criticality. But the main idea here is that this is far more manageable and realistic at the local level of each agent than at the global level of the whole complex system.In the next subsection we present both DREAM and AMOEBA the two collective artificial intelligence (Adaptive Multi-Agent Systems) based tools used in our second contribution in the semantic interoperability level.

2.5 Interoperability based Dynamic Data Mediation using Adaptive Multi-Agent Systems

Both DREAM and AMOEBA rely on a bio-inspired collective artificial intelligence (Adaptive Multi-Agent Systems), defined in our [Motie et al., 2018] as follows:

2.5.1 Multi-Agent Systems

A Multi-Agent System (MAS) [Weiß, 1999] is defined as a system composed of autonomous agents which pursue individual objectives and which interact in a common environment to solve a common task. The autonomy of an agent is a fundamental characteristic: an agent is capable of reacting to its environment and acts from its own decision, relying only on a limited and localized knowledge of the environment, using a set of skills and tools.

2.5.2 Self-Adaptive Multi-Agent Systems

A self-Adaptive Multi-Agent System (AMAS) is a MAS able to adapt itself: adjust itself, organize itself, heal itself, etc. [Di Marzo Serugendo et al., 2011] to remain in a well-functioning state after a perturbation. It relies on the fact that the agents do not follow a global directive towards the solving of the problem but collectively build this solving. This produces an emergent [Corning, 2012] problem solving process.

2.6 Dynamic Data Relation Extraction using AMAS

DREAM[40] [Belghache et al., 2017] is an adaptive multi-agent system that extracts relations between data streams, on the fly, based on their dynamics correlation. DREAM is able to find data streams that evolve in the same way, even if there is a time delay between them, thanks to a new correlation metric called *Dynamics Correlation*, which allows to study (perceive and evaluate) data dynamics[41] by combining a conventional statistical analytics tool and a new physical analytics tool.

DREAM is an unsupervised learner that discover patterns (relations) in data, The set of these patterns are called "data relations model". The data relations model is represented by a dynamic graph as shown in the snap (cf. Figure 33), wherein a node expresses a data stream and an edge exhibits a dynamics correlation between two data sources, built and updated by the agents with additional cooperative behaviors.

[40] see reference 34

[41] Dynamics means the behavior or the evolution, in other words changes occurring over time

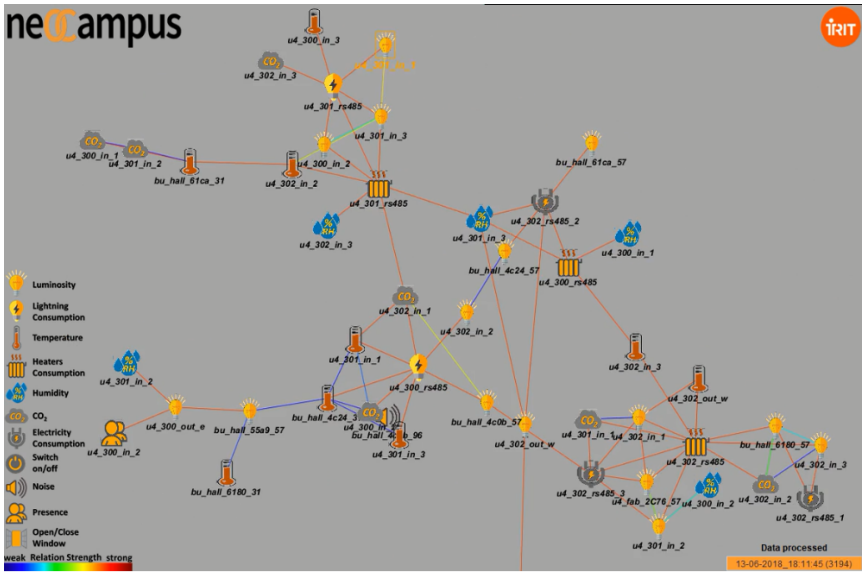

Figure 33: Data relation graph

Where:

- The nodes are created and destroyed by the percept agents
- The edges are managed by the relation agents
- The color a of an edge represents the strength of the correlation (cold/blue = weak, hot/red=strong)

We distinguish two concepts: percept and correlation where:

Percept

- handle one given data stream/source
- Receives and normalize the data
- Builds the phase space
- Send it to its Correlation agents
- Links itself to other Percepts creates common Correlation agents
- Helps other percepts to find dynamics correlations between them

Correlation

- Links two Percepts
- Updates the Local PSS
- Updates the correlation coefficient
- Detects and screens the Situations of Interest (SI)

Dreams behavior

- 1. Initially, each data stream is subscribed
  - i. Some percepts are created
- 2. Each percept choose randomly a stream to handle
- 3. A Percept first builds a random neighborhood
- 4. The Correlation agents sends SIs back to their percepts
- 5. The percepts spread it through their neighbors if the SI is a dynamics correlation
- 6. Otherwise, percepts with well-defined dynamic seek correlation with others
- 7. If, the Correlation agent doesn't find any SI, it becomes useless and signals it to its Percepts. Then the agent destroys itself
- 8. Likewise, when a Percept doesn't receive new SI, it expands its neighborhood randomly to find new correlations
  - i. If it doesn't work the Percept leaves the stream and moves on to another one. First randomly, then it will aim oldest stream

the statistical analytics tool is an incremental version of Pearson's correlation coefficient r [Cooper, 2014], defined as follows:

$$r(A,B) = \frac{\sum_{i=1}^{n}(a_i b_i) - n\bar{A}\bar{B}}{n\sigma_A\sigma_B} \tag{2.1}$$

$$\bar{A}_i = \frac{S_i}{i} \qquad with \quad S_i = S_{i-1} + A_i, \quad S_0 = 0 \tag{2.2}$$

$$\sigma_{A_i} = \frac{Q_i}{i} - \bar{A}_i^2 \;\; with \;\; Q_i = Q_{i-1} + A_i^2, \;\; Q_0 = 0 \tag{2.3}$$

Where,

- A, B are two variables (data features).
- $\bar{X}$ is the mean of X.
- σ_X is the standard deviation of X.
- n is the number of data points (values).

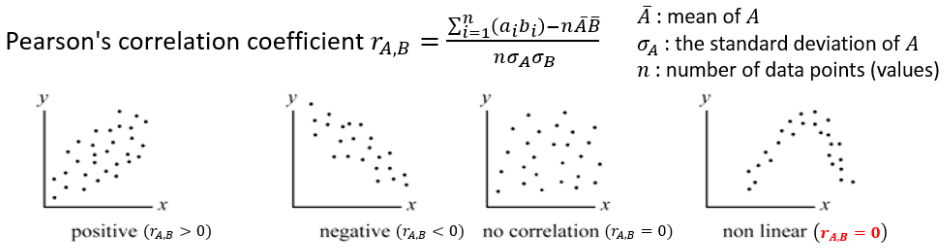

Figure 34: Pearson's correlation plot

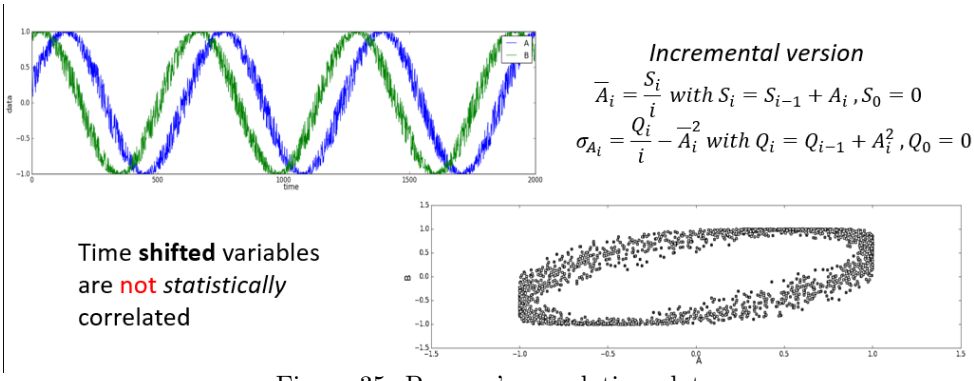

Figure 35: Pearson's correlation plot

This first analytical tool tells us how one variable behave depending on another. In order words how one variable is correlated to another This correlation can be seen thanks to the "scatter plot", see (cf. Figure 34) and is evaluated with the "Pearson's correlation".

If coef > 0, A and B are positively correlated (A's values increase as B's). The higher, the stronger correlation. If coef < 0: A and B are negatively correlated. If coef $= 0$: they are independent. This coefficient as it is formulated needs all the data from the start in order to be computed. So, in order to fit this coefficient in our real time system [quick reply with no data archive/memory] we use an Incremental version see (cf. Figure 35), wherein the mean and the standard deviation are updated each time new data come.

The correlation coefficient has one major issue: it considers two non-linearly related variables as not correlated. Thus the correlation coefficient misses complex correlations like two similar variables with a time shift between them. Therefore, we need another analytical tool able to do so.

The new physical analytics tool, called *Phase Space Similarity* metric see (cf. Figure 36), is a comparison between two data streams dynamics (evolution) using their dynamics abstraction, the *Phase Space*. This metric is given with the following equations:

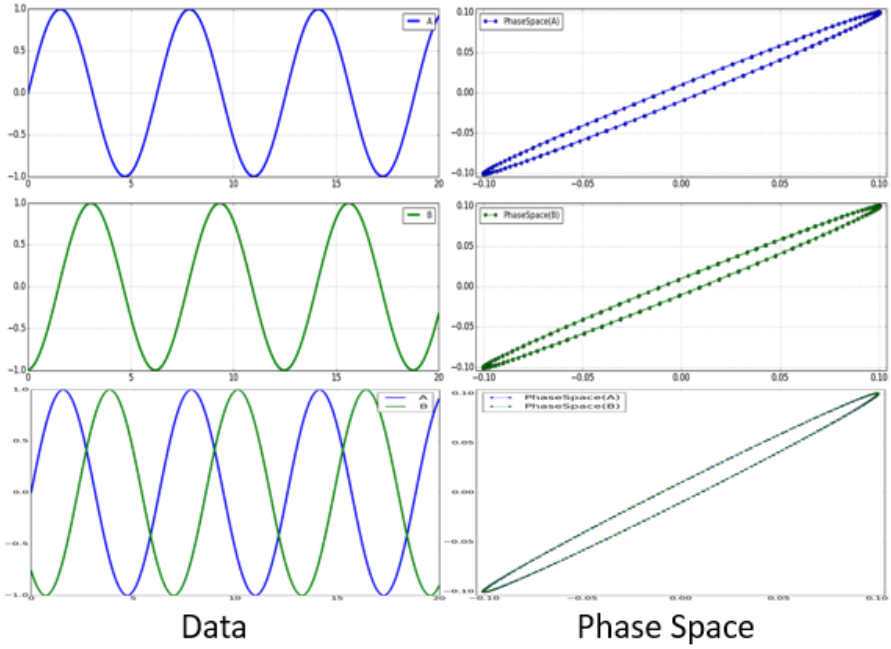

Figure 36: Phase space similarity

$$(psx_{A_i}, psy_{A_i}) = (A_i - A_{i-1}, A_{i+1} - A_i) \tag{2.4}$$

$$PS_A = \{(psx_{A_i}, psy_{A_i}), \forall i \in [1, n-1]\} \tag{2.5}$$

$$LPSD(A_i, B_i) = e^{-0.5\left(\sqrt{|psx_{A_i} - psx_{B_i}|} - \sqrt{|psy_{A_i} - psy_{B_i}|}\right)^2} \tag{2.6}$$

$$LPSS(A_i, B_i, m) = \frac{\sum_{j=i-m+1}^{i} LPSD(A_j, B_j)^2}{m}, m \geq 1 \tag{2.7}$$

The *Dynamics Correlation* metric, see (cf. Figure 37), is combination of the two previous metrics, Phase Space Similarity (LPSS) for detecting situation of interest SI and partial Pearson's coefficient (r) to screen false positive correlations, according to the following procedure:

For a system of n inputs, it takes $\frac{n(n-1)}{2}$ calls of the dynamics correlation tool to examine all the possible relations, which corresponds to a temporal complexity of $O(n^2)$ if computed sequentially or a spatial complexity if computed in the same time, in either ways, this high complexity prevents the system to scale up.

Thence, for the sake of designing a component that produces a dynamic graph model of the data relations on the fly, DREAM incorporates the dynamics correlation into an AMAS to focus only on the most probably correlated data and therefore reduces the computing power and the time to find all the data relations.

Algorithm 1: Dynamics Correlation

if $LPSS^2 \geq 0.7$ *and Partial* $r^2 \geq 0.01$ **then**
  return $Max(LPSS^2, r^2)$; //it is a true Dynamics Correlation
if $LPSS^2 \geq 0.7$ *and Partial* $r^2 < 0.01$ **then**
  return *-1*; //it is a false Dynamics Correlation
if $LPSS^2 \geq 0.7$ *and Partial* $r^2 \geq 0.01$ **then**
  return *0*; //it is not a Dynamics Correlation

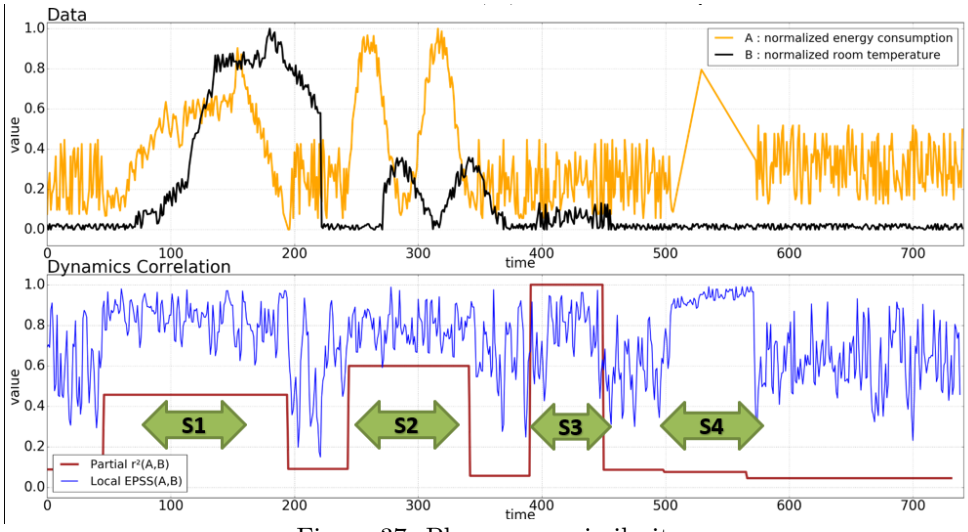

Figure 37: Phase space similarity

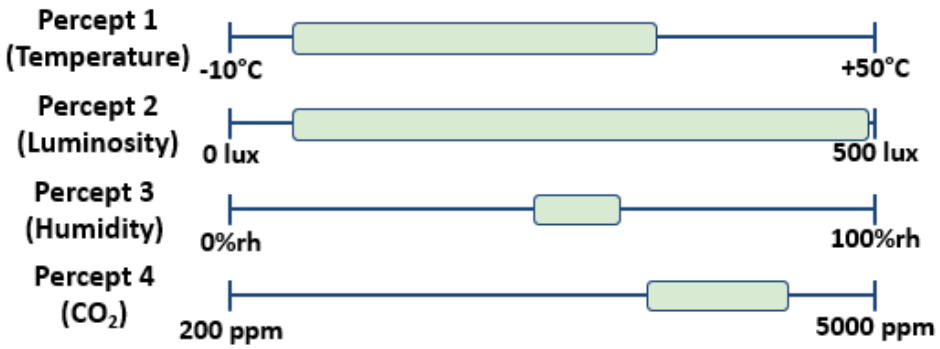

Figure 38: A snapshot of one Context agent at a given time T (its validity ranges)

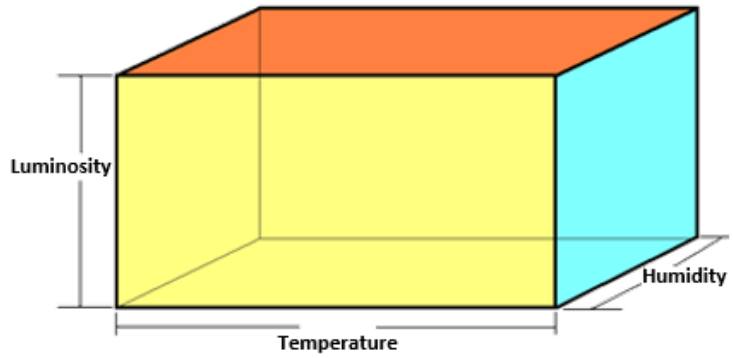

Figure 39: Hyperrectangle visualization of one Context agent with 3 percepts

2.7 Agnostic MOdEl Builder by self-Adaptation

AMOEBA[42] [Verstaevel et al., 2017] is composed of agents that build an agnostic (without knowledge of the meaning) model of AMOEBA inputs, or *"percepts"*, on the fly. These agents, called *contexts*, can be summarized as sets of percepts data ranges wherein each data point from one percept range corresponds to the data in all the other percept ranges (see figure40-a). All the context agents, see (cf. Figure 38), constitute a mapping of co-occurring data in the whole data space (see figure40-b). Moreover, a context agent has a confidence proportional to its percepts data co-occurring frequency. A simple way to visualize this structure is to represent the context of a Context agent such as n-orthotope (or hyperrectangle) see (cf. Figure 39)

We use AMOEBA, as an unsupervised learner[43], to provide a data translation function (described in 3.2). Consequently, the more inputs and data AMOEBA gets the higher is the confidence in the model accuracy. This translation function provided by AMOEBA insures the semantical

[42] see reference 35
[43] Learns by itself without an expert telling the correctness of the learning result (feedback)

interoperability of the data dynamically correlated by DREAM. This process is explained in the next section.

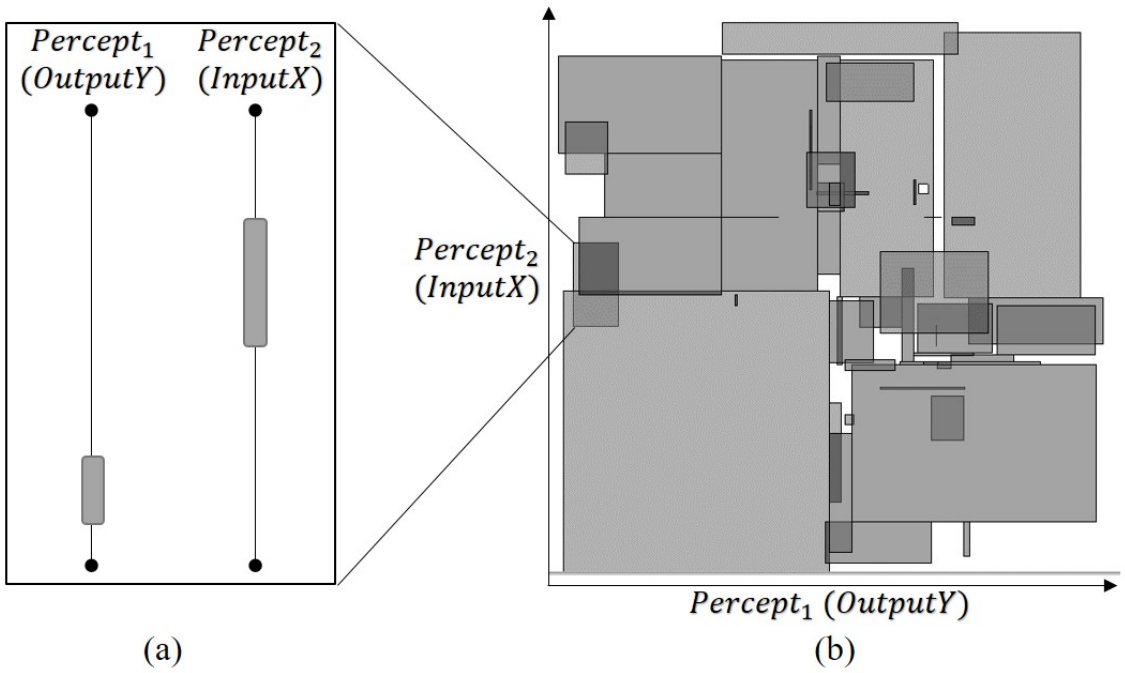

Figure 40: An example of AMOEBA context agents. (a) A context agent and its inputs ranges. (b) The mapping of a two dimensional data spaces (graphical representation of the context agents)

3 Dynamic Data Mediation

In our previous approach, we set up an individual mediator for each component, using a context mediation, to make the subsystems understand each others data. However, the creation and maintenance of such mediation model, as a standard used by all the subsystems, requires considerable amount of time and efforts. This mediation model leads to define three types of integration rules:

- constraint rules that reduce the objects to be considered according to predicates,
- merge rules that aggregate instances of classes of similar,
- and join rules that combine information from multiple object classes based on one or more common properties.

We distinguished two types of mediation:

- Schema mediation which provides better extensibility and often better scalability (object interfaces, rule-based language).
- Context mediation seeks to discover data that is semantically close, it is able to locate and adapt information to ensure complete transparency. We can therefore take advantage of the

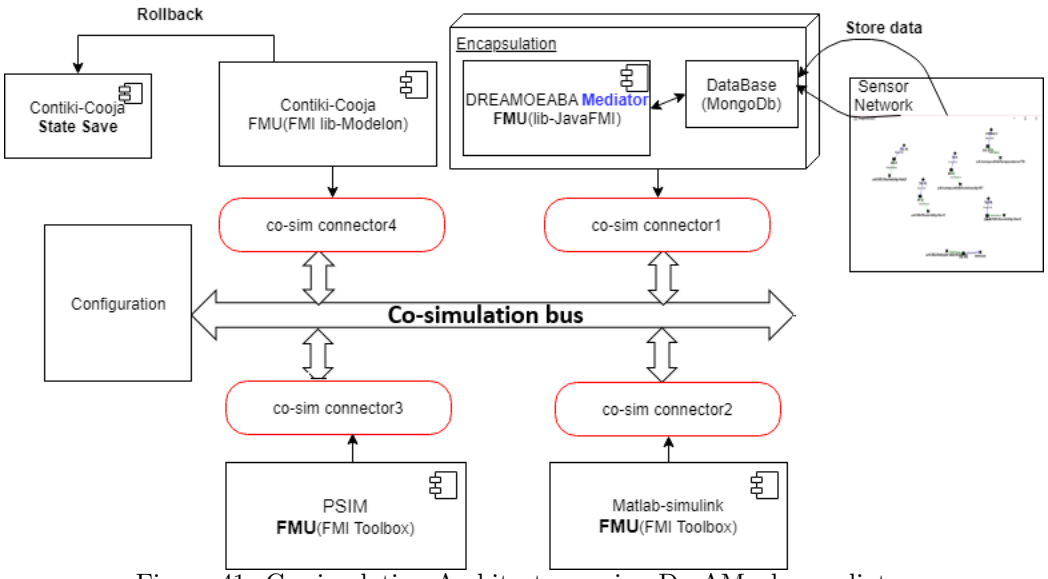

Figure 41: Co-simulation Architecture using DreAMoeba mediator

robustness of the mediation schema approach and combine it with the semantic approximation techniques of context mediation. This semantic mediation will be used to correlate, aggregate and dispatch data with respect to the control that we want to enforce on data produced or consumed.

Therefore, we extend our co-simulation framework with a new component for dynamic data mediation, called *"DreAMoeba"* (see figure 41), operating as follows:

- Use DREAM to link the subsystems dynamically, each link is described with an *(Output, Input)* couple (3.1).
- For each data couple query, use AMOEBA to translate the data form *Output* to *Input* (3.2).

3.1 Dynamic Subsystem-to-Subsystem Graph Model

A *Subsystem-to-Subsystem Graph Model (SSGM)* is a connection graph between the subsystems, encapsulated in FMUs, which displays their communications bridges and therefore describes which outputs from one subsystem are connected to the inputs of another.

When one wants to start a co-simulation, he has to describe manually SSGM at the composition step. This is a hard and time consuming task considering, in the one hand, that it requires some expert-domain knowledge about the data and, in the other hand, when dealing with real large cyberphysical environments the number of available data is high. Moreover, once this SSGM is established, it can't be modified during the simulation step for the sake of an interactive and dynamic integration of a new subsystem. This requires a shut down of the system and then an update of the SSGM with new subsystems and finally a restart of the co-simulation.

Hereafter, we describe how the use of DREAM can reduce, and ultimately get ride of, the human interaction with the co-simulation framework. For a better understanding, we virtually split DREAM's behavior in two.

3.1.1 Initialization Behavior

All the salves inputs and outputs (figure 42-a) are considered as input data streams for DREAM which builds the dynamics correlations graph for the data streams (figure 42-b). In addition, we extend DREAM's agents with a filtering behavior in order to discard the links (dynamics correlations) between the data streams of the same subsystem and links between data streams of the same type (input-input, output-output), because such links are irrelevant to build a subsystem-to-subsystem graph model (figure 42-b). Besides, we leave to the users the choice, if desired, of a specific subsystem to which they want to link, resulting in *the subsystem-to-subsystem graph model* (figure 42-c).

3.1.2 Nominal Behavior

DREAM is able to handle large amounts of data streams leading to a huge number of links and consequently to ({output1, output2... outputN}-input) couples. In such situations, where one input can be linked to several outputs, DREAM keeps only the strongest link (link with the highest Dynamics Correlation metric value). As you can see in figure 42, the input *InB3* is linked with *OutA2*, *OutB1*, *OutC1* and *OutC2*. However, DREAM discards (*OutB1-InB3*) because they are of the same subsystem *B*. Moreover, DREAM discards (*OutA2-InB3*) and (*OutC2-InB3*) as they are weaker than (*OutC1-InB3*) regarding their Dynamics Correlation strengths.

In addition, DREAM is self-adaptive to its environment (see 2.6). In other words, it can update the SSGM on-the-fly. This continuous adaptation allows our co-simulation framework to cope with changes occurring in the cyberphysical environment. These changes are:

- the evolution of the existing subsystems inputs and outputs. DREAM removes links between inputs and outputs of different subsystems that are not dynamically correlated anymore and links the newly dynamically correlated ones.
- the arrival of new subsystems. DREAM provides, for the new subsystems, links with existing subsystems and replaces previous links with stronger ones, if found.
- and most importantly, the departure of an already present subsystem in the co-simulation. In this case, DREAM finds, if they exist, other relations to replace the links destroyed by the subsystem departure from the co-simulation.

3.2 Data Translation using AMOEBA

By virtue of its genericity[44], DREAM doesn't process high-level semantics i.e. it doesn't handle the meaning of the data like temperature, luminosity, CO2, energy consumption, etc. Hence, it finds heterogeneous links (luminosity-temperature, humidity-CO2...) which is problematic knowing that data can have different domains and different semantics.

[44]Ability to be applied on any application domain data

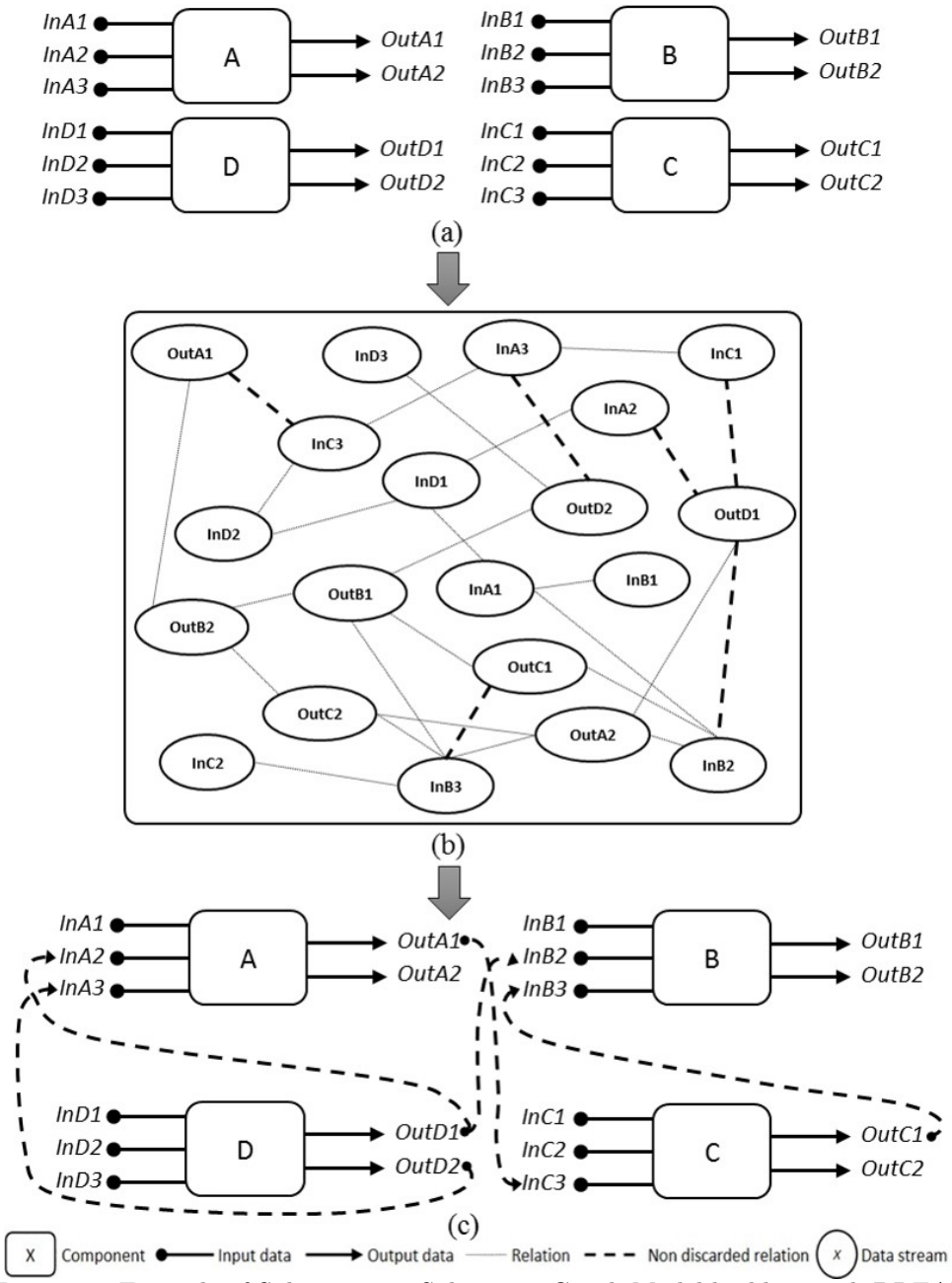

Figure 42: Example of Subsystem-to-Subsystem Graph Model building with DREAM

Thus, we combine DREAM with a second self-adaptive artificial intelligence, called *"AMOEBA"* (see 2.7), able to act as a translator between heterogeneous data couples *(Output, Input)*. This translator is a matching function between the *Output* data and the most probable *Input* data. Ideally, we would have a homogeneous mapping of the data space such that there are context agents, with only one data point per each percept (AMOEBA input) range, that cover all the data space. However in cyberphysical environments some data are missing or noisy, which leads to heterogeneous mapping with overlapping context agents of different sizes (see figure40-b). Therefore, AMOEBA, see (see figure 43), has to reduce the set of the contexts that map the *Output* data, using if needed, the other percepts data and their confidence, then producing the *Input* data by returning the mean value of all the remaining context *Input* ranges, as follows:

Algorithm 2: Contexts based Translation Function

```
1  Contexts_set = {all the available context agents that map the data space and have a
     percept data range containing the Output data} ;
2  if ||Contexts_set|| > 1 then
3      Contexts_set = {the context agents in Contexts_set that have at least one another
         percept, besides Output, data range containing the data available in the co-simulation
         bus at this step of co-simulation} ;
4      if ||Contexts_set|| > 1 then
5          for each context in Contexts_set do
6              if the context confidence is low then
7                  Contexts_set = Contexts_set − {context};
8  Input_data_range = {} ;
9  for each context in Contexts_set do
10     Input_data_range = Input_data_range∪ {context Input data range} ;
11 return the mean value of Input_data_range ;
```

3.3 Model Calibration

As we described it in subsection 3.1, DREAM continuously updates the links of the SSGM in order to keep only the best ones. These updates are more frequent during DREAM's initialization phase until it stabilizes. For this reason, we need a model calibration phase to generate a first more stable SSGM before introducing the subsystems in the co-simulation. Similarly, AMOEBA requires a calibration in order to provide a translator model used for the dynamic data mediation.

This model calibration, can be seen as the training of a machine learner. Thus, we need a database containing, for each subsystem, the data used by its input and provided by its outputs during a test run, to generate a first SSGM and translator model then start the co-simulation with them.

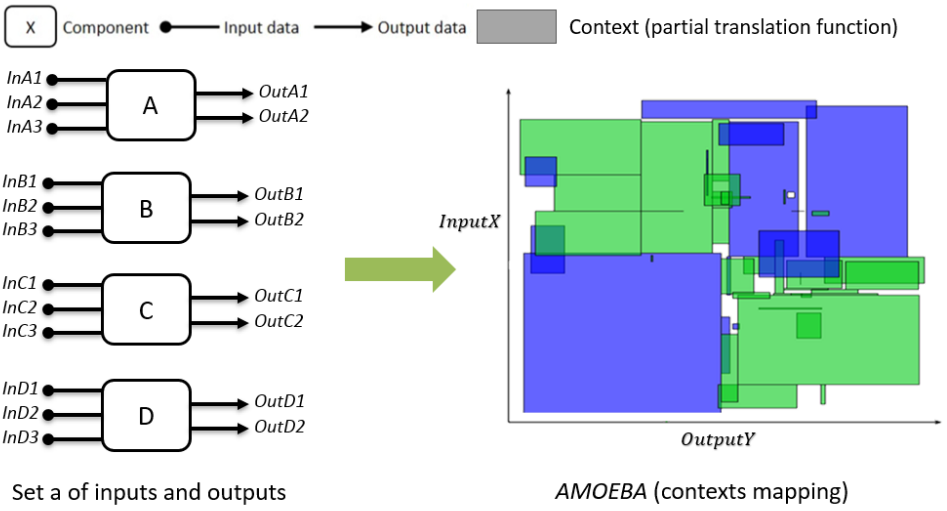

Figure 43: AMOEBA learns the data combinations

3.4 neOCampus Use Case

In this subsection as in 3.2.1 we took the same simulators used in 1.3.1. As the **collection of sensors data** is stored in a NoSQL **database** (mongodb), we use them as a training set for DreAMoeba which is a **java simulation** made to mediate data. It requires data provided from the other components to make correlations and translations then, provides the right dynamically linked data couples, as described in section 3 and illustrated with a use case scenario (see 3.5). We have also built a knowledge base from sensors data that provides real-time data and relationship between them in order to build later an ontology on which to map the data provided.

3.5 DreAMoeba Mediator

One of the problems encountered is the mediation part, since we want to achieve a semantic interoperability we used DreAMoeba (see 3). In order to grasp how it works, let's take the following neOCampus co-simulation environment:

- a room equipped with several sensors (luminosity, humidity, co2, presence detection, heaters energy consumption).
- Matlab Simulink (see 1.3.1) for optimizing the temperature and electric power consumption in the room.

As described in 3.3, we first calibrate DreAMoeba, which leads to discover the *(room luminosity, Matlab Simulink temperature input)* and *(room heaters energy consumption, Matlab Simulink electric power input)* couples. The latter couple is quite understandable since its *Output* and *Input* are

about the same entity, the electric power used for heating the room. However, the former is less understandable considering, at first glance, their semantic dissimilarity. Nonetheless, DreAMoeba analyses the data deeper than what human hypothesis allow by studying their dynamics correlation mainly through their *phase spaces*[45]. So, as you can see in figure see (cf. Figure 44) the two phase spaces are similar, meaning the luminosity and the temperature behave fairly in the same way.

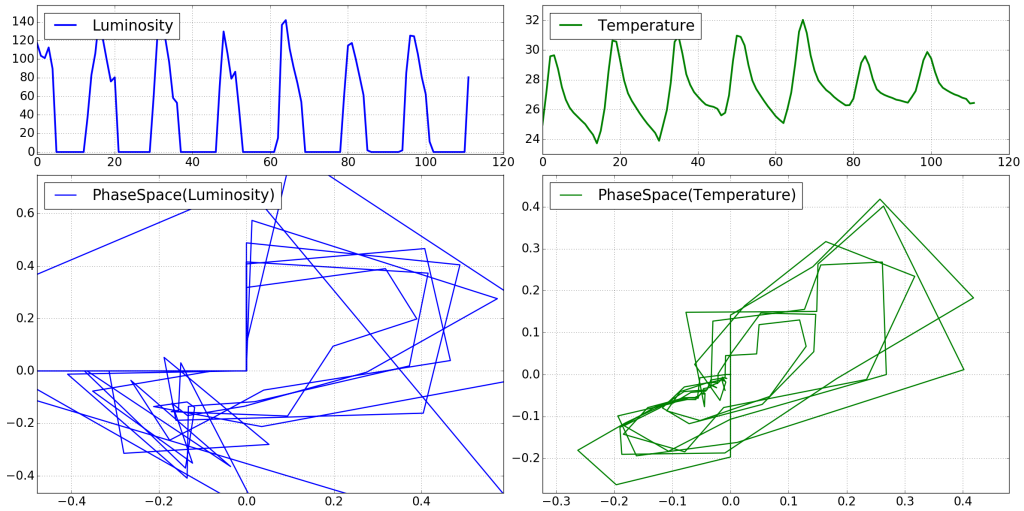

Figure 44: Luminosity and temperature data recorded by neOCampus sensors during one week.

In the neOCampus operation, the phase spaces make it possible to find correlations between heterogeneous data provided from the different sensors. The phase space similarity is the squared complement to 1 of the mean Euclidean distance between each couple of points taken from each phase space. As you can see in (cf. Figure 44) it focuses on the behavior of a single variable over time (time-independent representation of the data stream). So as the time shifted variables "temperature and luminosity" behave the same way they have the same phase space representation and are dynamically correlated. It compensate and correct the pearson's correlation coefficient issue which considers two non-linearly related variables as not correlated.

So in our case where components provide and require data, in some cases we can, for one component requiring temperature, send him luminosity instead (after translating it with Amoeba). This makes it possible to achieve a dynamic interoperability while interacting with the data.

The data translation (3.2) insured by AMOEBA for the (room luminosity, Matlab Simulink temperature input) is described in example algorithm 3.

[45]Representations that exhibit the evolution behavior of each data stream

Algorithm 3: Example of the data translation on the toy neOCampus use case

```
1 if Luminosity ≥ 20 then
2 |   AMOEBA returns a unique temperature as a result of the homogeneous mapping of the
  |     data space when Luminosity ≥ 20.;
3 |   In other words, each Luminosity data ≥ 20 corresponds to a single temperature data ;
4 if Luminosity < 20 then
5 |   Several temperatures are possible and according to the data translation function (3.2)
  |     AMOEBA return the mean temperature ;
```

For the sake of a more accurate translation, we add to AMOEBA one or more new percepts from the data Outputs available in the co-simulation environment. For example, by adding only the *humidity* percept to AMOEBA, it creates a third dimension to the data space, which allows AMOEBA to reduce the number of *contexts* that cover the *temperature* data and thus AMOEBA returns only one *temperature* data.

In the neOCampus operation since DREAM, and because of its genericity, does not deal with high-level semantics, that is, it does not handle the meaning of data we exploit the mechanisms of AMOEBA which was used as an unsupervised learner to arrive at our translation function without taking account of any oracle. So for the case where one component requires temperature which is not available, and DREAM found a dynamic correlation with luminosity provided with another component. We can thanks to AMOEBA make a translation from 'lux' to 'celsius' so the to components can exchange their data.

Let's take an example of two components in our co-simulation framework, the first one provide a X output value of temperature and the second required a Z Input value. AMOEBA needs to translate (X,Z) couples discovered by DREAM. Each context is a sub translations function, Cross-breed contexts to translate X to Z. Let's take X=20C.

Therefore, AMOEBA must reduce the set of contexts that map the output data, using other perceived data and their confidence, if necessary, and then producing the input data by returning the average value of all the remaining contexts . Retrieve the validity range of the context for the percept to which I want to translate. Either the range of validity is a point I return it see (cf. Figure 45).

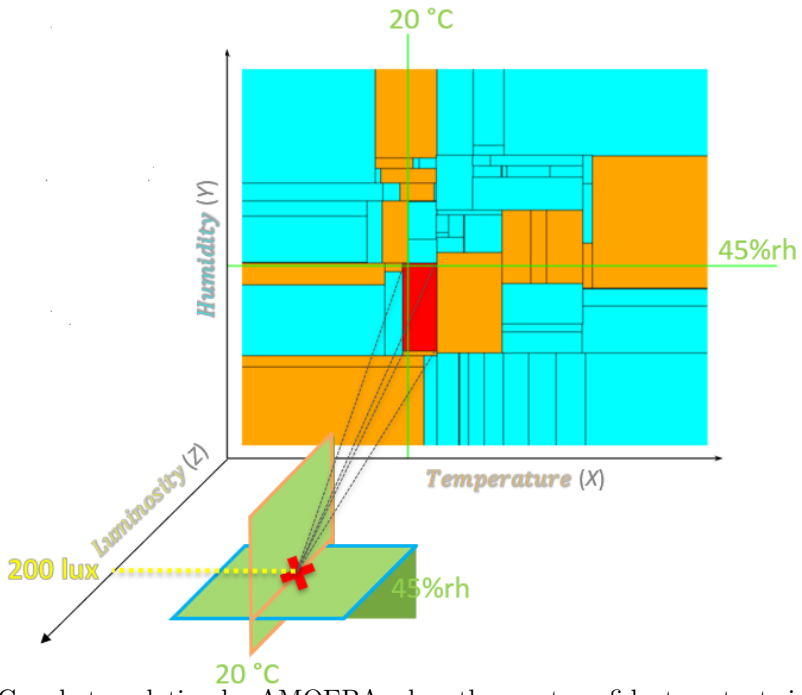

Figure 45: Couple translation by AMOEBA when the most confident contexts is one value

If there are several contexts that satisfy the intersection see (cf. Figure 46). I return the mean of the intersection of the validity ranges for the percept to which we want to translate (otherwise if the intersection is null, situation of non-cooperation could return to the old confidence.

- 1. Select contexts which validity range of the percept X contains the data X=20C
  - –> first hyperplan of X in Z dimension (X=20C)
- 2. If contexts>1
  - i. use another correlated couple (Y,Z) to reduce contexts –> second hyperplan of Y in Z dimension (Y= 45%rh)
  - ii. remove contexts which do not satisfy the intersection of the hyperplans
  - iii if there is more correlated couples go to 2
- 3. if the intersection of the validity ranges related to Z of the most confident contexts is one value –> return the intersection value see (cf. Figure 45)
- 4. if the intersection of the validity ranges related to Z of the most confident contexts is a range –> return the mean value of the range see (cf. Figure 46)

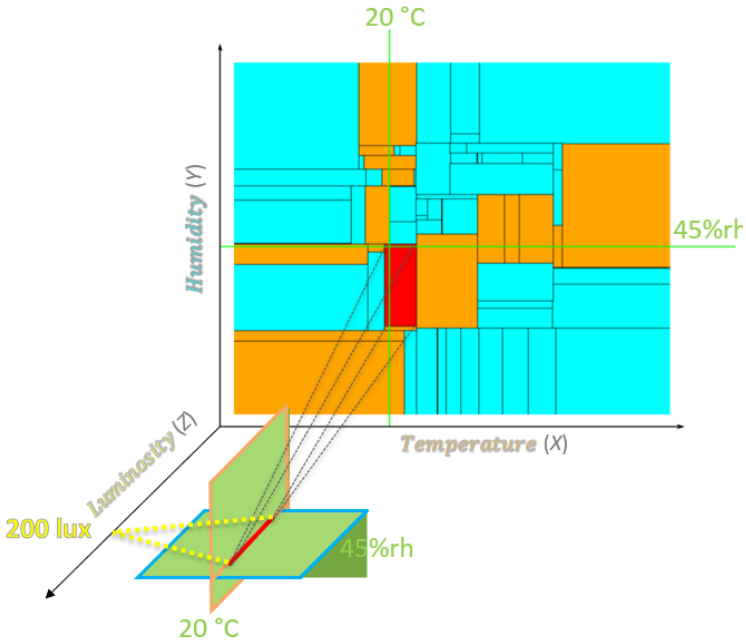

Figure 46: Couple translation by AMOEBA when the most confident contexts is a range

To sum up, the more AMOEBA has percepts the higher is the data space dimension, that leads to less overlapping *context* agents and in consequence the more accurate is the data mediation.

Our architecture has been implemented and our modeling works well: we took as example the 4 simulators and used FMI for co-simulation in order to generate a slaves (FMUs) and solve the structural problems. Our DreAMoeba component which is used to ensure the mediation, in order to achieve the semantic interoperability, was encapsulated using JavaFMI.

DREAM, as a gray-box, offers openness and self-healing features (make new connections and replace lost connections on the fly) to our co-simulation framework, when combined with an self-adaptive data translator will provide autonomous semantic interoperability.

Furthermore, these new features, without taking into account the semantic aspect through the data translation, can be applied independently to improve the current tools relying on FMI by (semi-)automatize the linking process between the slaves inputs and outputs.

While our framework was conceived for users we had to convince them to participate to the exercise to explain for them how they can couple their simulation tools allowing them to interact with the complex system in order to retain their business expertise and continue to use their own digital tools. The interest regarding co-simulation, is not only triggered by the coupling of the environments but also by the potential efficiency gain of decoupling a large system model. This

is exemplified in [Hippmann et al., 2005] where a model of an engine is split into subsystems, leading to a decrease of the simulation time by an order of magnitude.The idea of our approach as described in 1 is that tools generate and exchange models that conform to the FMI specification, called FMUs (Functional Mock-up Units) 3.1.5.

The problem is that the transition from the subsystem studied to an FMU component (black box) requires knowledge of the FMI standard, thus constituting an obstacle for our designers, inhibiting thereby their collaborations.
Rather than forcing the users to change their behavior to accommodate our framework, we will try, as described in the next chapter 1, to optimize the framework's structural interoperability level around users needs.

CHAPTER V: Conception Methods for and with the user

1 Conception Methods for and with the user

1.1 Problematic

Inspired by the user-centered design approach [Abras et al., 2004] see (cf. Figure 47), we studied components generation methods and proposed a prototype-based on the generation tasks to be performed - for partial automation. The idea is to allow the designers to preserve their tools, their favorite languages and their expertise in order to guide them for the co-simulation first step with other heterogeneous simulators.

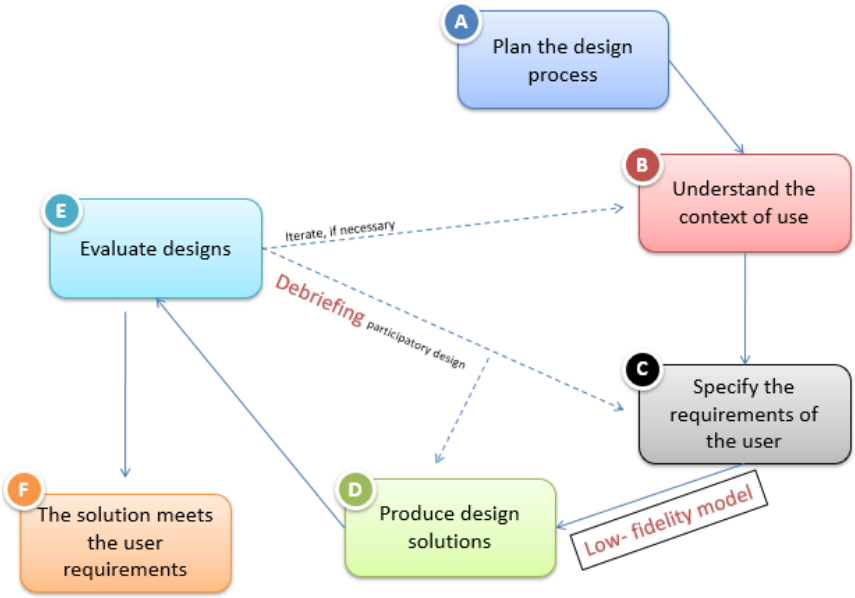

Figure 47: Our user-centered design approach

Designers working in the neOCampus project mostly use commercial off-the-shelf (COTS) simulation software [Boer and Verbraeck, 2003] to build and test their simulation models. Integrating these models to form a single meta-model is a major issue especially when distributed simulation technologies are not anchored in these software [Taylor et al., 2006].
We have seen that because of the lack of communication and collaboration between these different designers, their models were built completely in a disconnected way and do not benefit from the exchange of information that can simplify and accelerate their work.
Beyond these practices, another problem identified concerns the difficulty of being able to generate the co-simulation components (essentially components implementing the FMI) from the different simulators used. Indeed, the majority of the designers are experts in their field but 1/ are not necessarily computer experts and 2/ if they are, not often expert in co-simulation nor in FMI.

Convincing users in neOCampus project to participate to the exercise wasn't an easy task, we had to explain for them how they can couple their simulation tools, how it benefits them, allowing them to interact with the complex system in order to retain their business expertise and continue to use their own digital tools. to provide the right data for those who need it in real time and to let This time of taking over the FMI technology (time and practice) has led us to propose a mechanism to support the generation of FMU components based on user practices. This interface is intended to be easy to use for novice users, adapted and allows to accompany the process of generation of components ready to connect to our platform.

The problem is that the transition from the subsystem studied to an FMU component (black box) requires knowledge of the FMI standard, thus constituting an obstacle for our designers, inhibiting thereby their collaborations.
Rather than forcing the users to change their behavior to accommodate our framework, we tried, as described in the next section 1, to optimize the framework's structural interoperability level around how users can, want, or need to use it. Inspired by the user-centered design approach [Abras et al., 2004] see (cf. Figure 47), we studied components generation methods and proposed a prototype-based on the generation tasks to be performed - for partial automation.

The goal was not only the generation of pluggable components but we had rather a vision to embed them in the process and stay tuned to them, something that started well with the establishment of a prototype-based on the components generation tasks as the first need expressed by our users/collaborators. There were two phases:

- The pre-co-simulation where the need was to establish a user-centered process to facilitate the structural interoperability

- The post-co-simulation phase which aims to visualize the exchange of data and make accessible, in real-time, for a user the data interacting with, thus follow the evolution of his system, analyze the semantic interoperability, this last phase would be considered rather as perspective and will not be treated in this work.

As you can see in (cf. Figure 48), which describes a brief summary of the different phases constituting the implementation of our framework. It highlights, obviously staying on the pre-cosimulation phase, the locations where the need for a user interface design is proven and claimed by the collaborators themselves. It distinguishes:

- The purely structural interoperability "in blue" where the simulators are encapsulated hiding the internals and protecting the integrity of the model by preventing users from setting the internal data of the component into an invalid or inconsistent state. In our case it helped us reduce system complexity, and thus increase robustness, by allowing our collaborators to limit the inter-dependencies between components. It results in the generation of a black-box simulation components called FMUs (simulation components compliant with FMI specification), limiting the data exchange to discrete communication points (knowing that the system model is solved by internal solver between these communication points)

- Its combination with semantic interoperability including of course our artificial intelligence layer combining our AMASs DREAM and AMOEBA "in red", in order to provide a dynamic mediation for adaptation of the data which allows to go further towards the development of global and open simulation environment. The data administration is supported thanks to the use of cosimate. Using a distributed architecture, where I / O procedures of each simulator take care of extracting its data and communicate them to the co-simulation bus used as a communication protocol.

To achieve the phases described in (cf. Figure 48) we had to do a study and design which concerns our original approach, an experimentation since people can do tests, an implementation as we can install it permanently, and then to perform a data collection. We distinguish:

- The structural interoperability where we set up the medium of connection using FMI standard which was implemented and the FMUs were then generated for each model in our neOCampus operation

- The semantic interoperability where we had to deal with management of the data passing through the medium. Two sub-types were defined:

  - a semantical interoperability using mediation for adaptation of the data - Raw Interoperability - as it deals mostly with the format of data

  - a semantical interoperability using dynamic data mediation based on AIs - Dynamic interoperability: Interaction - which is an important part in neOCampus as users can ask to interact with data that we can provide in the required form. As we can respond with the users required data for tests purposes for example.

- Involvement of designers (HCI) [Motie et al., 2018]. One of the main issue found in this work is to convince collaborators (experts in their domains) about the importance of co-simulation (to validate their simulators, get the required data provided by others) while we made a great effort to understand each system and explained to them in easy words how the co-simulation work we found many obstacles as we wanted to develop with the user so we:

  - Design scenarios

  - Prototype and develop a "Medium fidelity" Interface

  - Evaluated it

The first two frames at the top consist of two simulations that were encapsulated based on FMI standard described in 3.1 within FMUs using a centered design approach (HCI in green boxes); resulting from a reflection - removing an obstacle "knowledge of FMI" - involving the user in the co-simulation process through the generation of interactive components. A graphical interface

described in 1 was developed to guide the user in the generation process from the number of input and output variables, entering names and types of variables, respect of cast and spellchecker, until the download of the generated component, providing a summary of the performed tasks.

The frame at the bottom of the figure consist of the dynamic interoperability where we interact with the data using AMAS described in 2; Our DreAMoeba system described in 3.5 was then encapsulated in an FMU to be connected to our co-simulation framework and ensure the dynamic correlation and translation between components inputs and outputs.

Cosimate (plays the role of master algorithm) offers a complete simulation environment dynamically linking heterogeneous simulators and which can be extended to different simulation environments on different platforms described in 2.3. Cosimate provides librairies that control acces to its co-simulation bus and adapts to the network configuration, offering a co-simulation based on a multi-client multi-server to avoid unnecessary communications between simulators instantiating local routers for each computer in the co-simulation.

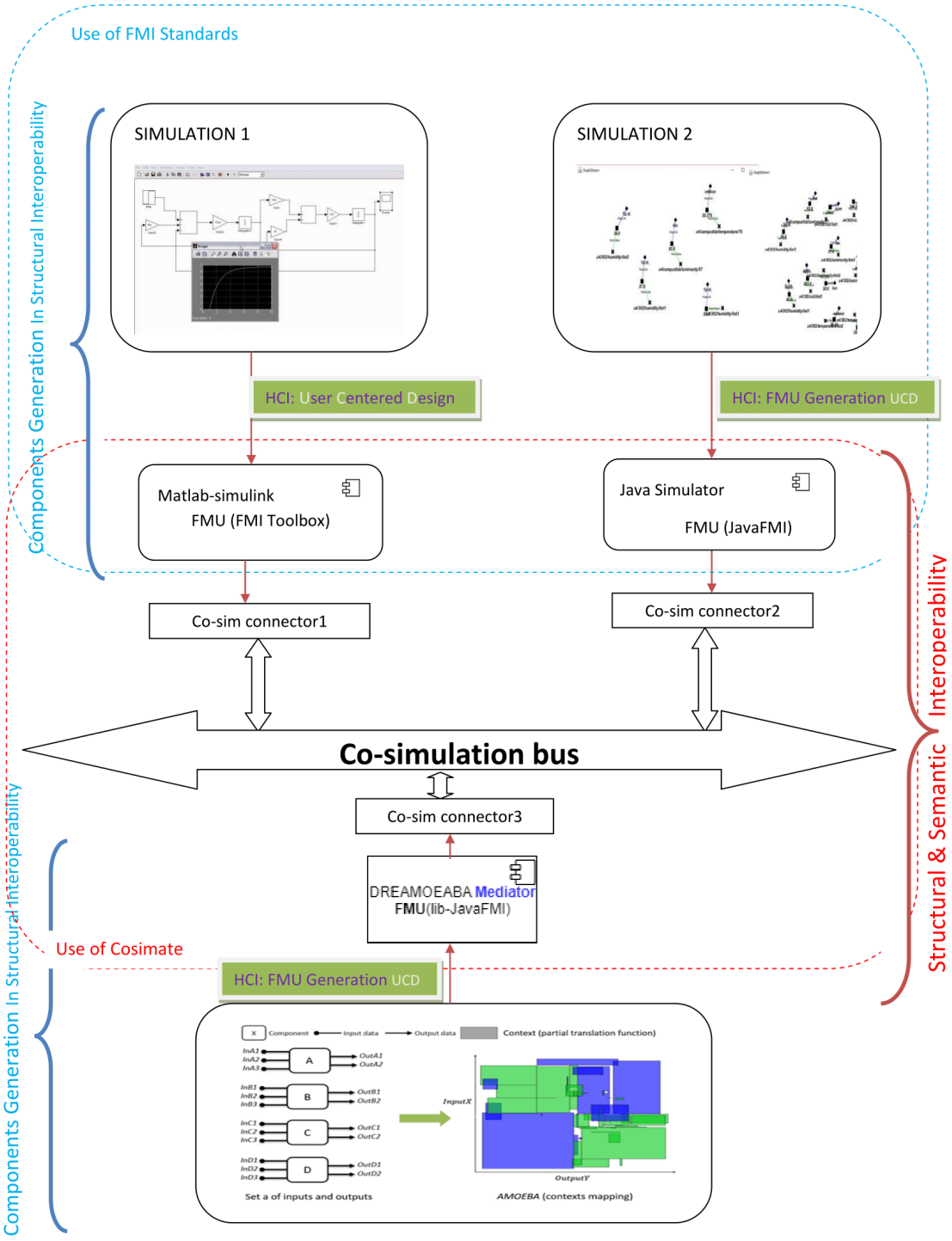

Figure 48: Our framework phases and HCI need

1.2 Accessible Interface design

Most of the guidelines used to consider accessibility in the design phase are best taken into account for the final product by the designers and checked at the end of the cycle. They do not necessarily meet the accessibility needs of users that are part of the design process itself. Understanding the needs of users is a necessity which is now largely achieved by the integration of these users into a participatory design process. As in the case of the Web Accessibility Initiative (WAI) providing guidelines, called the Web Content Accessibility Guidelines (WCAG) to help developers make their web sites accessible [Lazar et al., 2004].
Participatory design, as an interactive systems design process, is a way to obtain a better expression of the needs from the beginning of the design process, by refining the functional analyzes and specifying the specifications from the point of view of the use that will be made of the future interface. There are many methods and tools to implement such a process. Some basic rules for implementation are defined by ISO 13407 "Human-centered Design Process for Interactive Systems". In the area of computer systems design, better take into account the usage needs of users resonates with so-called user-centered design (UCD) approaches.

The ISO 9241-210 standard which was replaced by ISO 13407 defines five criteria for applying and implementing the approach:

- Taking account of users, their tasks and their environment upstream
- Active participation of users, ensuring the fidelity of needs and requirements related to their tasks
- The appropriate division of functions between users and technology
- Iteration of design solutions, up to the needs and requirements expressed by users
- The intervention of a multidisciplinary design team, aimed at an optimal user experience

The basic principle is to adopt anthropocentric design approaches to ensure the acceptability of the systems (from the point of view of their usefulness and from the point of view of their usability). This ergonomic evidence promotes design methods that more closely examine user activity and encourages designers to adhere to and to respect the system's compatibility with the needs of the user. Among other things, it is essential to achieve "active user participation" in all phases of our process that include:

- Analysis of user needs and activities

- Design
- Prototyping
- Evaluation of the designs in relation to the requirements

Some existing methods that can be used to implement this process are presented by Muller [Muller and Kuhn, 1993] and in ISO 16982 "Usability Methods for Human Operator-centered Design" [16982, 2002]

The participatory design cycle begins with the analysis of user needs and activities. The ISO 16982 standard offers, for example, methods of user observation, questionnaires, interviews, or the study of available documents.
To achieve this phase with designers like those of the neOCampus operation, one should understand their needs to co-simulate (need for data, need to validate their simulation,...), their working environments, the tools they use, and their levels of expertise with computer (softwares, FMIs, GUIs,...).
Often in a phase following this first, it is useful to put into practice methods of creativity, such as brainstorming [16982, 2002], to produce ideas for solutions. There are variants and more specialized methods like the "Group Elicitation Method" [Boy, 1997] which proposes "brainwriting", a written variant of brainstorming. In our case where it's difficult to bring together the different researchers working in neOCampus. We assumed that the brainwriting phase isn't that relevant since we collected the main needs of our users and make sure of the importance of the implementation of a black-box components generation tool, and especially to save the learning effort of the FMI standard. We went then directly to the design phase, then replace this brainstorming by adding a debriefing step as shown in (cf. Figure 47).
For the creation of solutions, the second and third phases of the process, there are many possibilities. The most common is to use low-fidelity prototypes. These are produced as in our case by us designers from the ideas generated collectively. They are used to present to users solutions to evaluate, validate or refute concepts or interactions, and to choose or propose new ideas. For the realization of these prototypes the designer has the choice between several methods. These are often based on the use of visual content. Rettig [Rettig, 1994] and Snyder [Snyder, 2003] show, for example, the use of paper prototyping, in which manipulative and discussion interfaces are prepared in the form of drawings or collages. The "Wizard of Oz" experiment [Kelley, 1984] proposes to simulate the interactive functioning of the final prototype. This methodology is often based on such a visual paper mock-up. Serrano [Serrano and Nigay, 2009] proposes with "Open Wizard" a software solution of magician of Oz for multimodal systems. It allows to simulate input modalities but does not allow the simulation of the output modalities.
An alternative to the Wizard of Oz is to code low-fidelity prototypes. According to Sefelin [Sefelin et al., 2003], the results achieved with these prototypes are equivalent to those obtained with paper models. In addition, the interviews conducted at the end of the tests comparing paper models and software prototypes reveal that 22 of the 24 subjects say they prefer working with software prototypes. New technologies like Adobe Flash or MS Silverlight make it easy to create low-fidelity prototypes. We coded a low-fidelity prototype which addressed two very interesting points:

- Adequacy between proposed functionalities and user needs
- Adequacy between the interface and the users

The evaluation phase consists in measuring the usability of the product, i.e to validate the satisfaction of the users in the realization of the evaluated tasks. Among the different methods of evaluation possible, the main one is the user test. This consists in placing the user in a situation of real use of the product and in observing the difficulties encountered. The evaluation makes it possible to identify the points to be improved on the model and thus to prepare the next version which will be tested again and so on see (cf. Figure 47). Experience shows that two to three iterations are usually sufficient to finalize the design of the interface.
During this last phase, we defined qualitative and quantitative measures, see 2 for the results part, to be satisfied such as:

- The success rate for completing the tasks to be performed
- The number of errors made (and possibly the workarounds performed)
- The execution time of each task
- The number of steps required to complete the task
- The possible recourse to a support or a help internal or external to the product (ex: we as much as animator of the test session)
- User satisfaction

User-centered design is mainly used in computer design and is based on ergonomics and usability criteria. This approach differs significantly from other design approaches in seeking to tailor the product (usually the user interface) to the end user rather than imposing on it a mode of use chosen by the designers.
User interfaces in modeling have traditionally followed the WIMP (Window, Icon, Menu, Pointer) paradigm. Though functional and very powerful, they can also be cumbersome and daunting to a novice user, and creating a complex model requires considerable expertise and effort.
Many tools exist to design interfaces around large, complex data sets, used by many researchers and companies to convey specific data in a clear and understandable manner. We can cite sketch-based interfaces for modeling (SBIM) [Olsen et al., 2009], allowing sketches (hasty freehand drawings) to be used in the modeling process, from rough model creation through to fine detail construction. Balsamiq [Guilizzoni, 2010] is a graphical tool created to draw user interfaces making it, using a software, easy to drag and position elements. Despite the fact that Balsamiq is an essential technique for successful design practice, it's far from being the most compelling way to communicate one's vision to the larger team.

Balsamiq was the graphical tool used as it simulates the experience of writing on a whiteboard, but using a software prototyping tools which helped us design iterate faster.

The improvement of an interface from an ergonomic point of view can be provided by several means:

- Those involving users, with performance indicators or more subjective (satisfaction / preference questionnaires, user tests, focus groups, task analyzes, card sorting...)

- Those solely based on the intervention of the ergonomist (expert inspection, heuristic evaluation, cognitive walkthrough, benchmarking for the Web or review of systems and similar products for software solutions)

As we want our tool to achieve its simplicity of use, its ease of learning, its use without errors and the satisfaction of its users, we distinguished two characteristics:

- Usefulness: the interface must be relevant to the objectives of the target user

- Usability: the extent to which a product can be used by identified users to achieve defined goals effectively, efficiently and satisfactorily in a specified usage context

The establishment of ergonomic criteria, which constitute the classification (typology) of the basic rules that condition the usability of an interface, aims primarily at two complementary objectives:

- evaluate the usability of a software by serving as a basis for the establishment of evaluation grids (checklist)

- take into account - from the initial stage of development - the ergonomic aspects of our tool (guide during the design)

In this context several researchers, C. Bastien, D. Scapin in [Bastien and Scapin, 1993], J-F. Nogier in [Nogier, 2008], H.X. Lin in [Lin et al., 1997], agree on the list and classification of these basic ergonomic criteria which are also at the origin of certain standards in the field. Other heuristics exist as well in the literature with a strong overlap that define basic rules to follow as in [Nielsen, 1999].
In order to realize our user interface, we were interested in the C. Bastien, D. Scapin criteria that have been the subject of experimental evaluations, and demonstrated that they were offering measurable advantages over the use of other references, and also provide a common framework and vocabulary.

1.3 Ergonomic criteria

These criteria, [Bastien and Scapin, 1993], can be reduced as follows (✓symbol designates that the criterion is taken into account during our conception):

- Compatibility: the ability of the software to integrate into the actual activity of users, reducing the transfer of knowledge between the business and the software. Some recommendations are to follow:
  - Speak the user's language (avoid computer jargon) ✓
  - Use familiar metaphors ✓
  - Arrange the elements of the interface according to the task of the user ✓
  - Access to functions must be compatible with the user's task ✓
- Guidance: includes all the means implemented to assist the user in the use of the software. This facilitates the use of the system and its learning. The user must understand the interactions that are expected of him by an operation that appears clearly. Broken down into four sub-criteria
  - Incitement: bringing together the means to lead the user to perform specific actions, providing the list of expected entries (drop-down lists, ...), clearly indicating the required fields, and giving the data entry format, encouraging the user to correctly enter the data, etc. ✓
  - Grouping/Distinction: consists in guiding the user by grouping information and functions of the same type (principle of similarity) and separating objects that are different, distinguishing the presentation format and the position in the interface ✓
  - Immediate feedback: combines all the elements used to show the user what the system is doing and providing feedback in response to each of their actions (increase user confidence). i.e report long treatments with an indication of waiting ✓
  - Readability: facilitate the perception of textual and iconographic information by a judicious choice of their properties and their disposition ✓
- Coherence: concerns the overall homogeneity of the human computer interface, distinguishing presentation (graphics, location, ...) and behavior (system reaction, messages, ...) ✓
- Adaptability: the ability of the interface to react and adapt according to the context and according to the needs and preferences of its users, broken down into two sub-criteria:

  - Flexibility: the means available to users to customize the interface to take into account preferences, skills, habits (drop-down menu, context menu, ...)

  - Taking into account the user experience: the means implemented to adapt to the different levels of user experience, knowing that the user experience may vary over time ✓

- Explicit control: concerns aspects related to the degree of control that the user has over the treatments performed by the interactive system, broken down into two sub-criteria:

  - Explicit actions: the relationship between the user's actions and the processing that will be performed by the application in response to those actions. (i.e do not trigger operations without the explicit consent of the user, etc.) ✓

  - User control: Regarding the fact that the user must always have control over the system and control the operations and their progress (interrupt, resume) ✓

- Error management: groups together various ways to avoid or reduce the user's errors and, if necessary, to correct them in order to maintain the integrity of the system broken down into three sub-criteria:

  - Protection against errors (prevent the user from committing them) ✓

  - Quality of error messages (clearly inform the user) ✓

  - Correction of errors (allow him to correct them) ✓

- Workload: minimizing both the amount of information the user needs to take into account and the number of elementary actions he / she must perform to accomplish a given task, broken down into two sub-criteria:

  - Brevity: concise, and minimal action that aims to minimize the number of actions needed to achieve a goal, to accomplish a task ✓

  - Informational density: delete all items that are not directly related to the current task and could unnecessarily distract users ✓

- Meaning of codes and denominations: It is not enough to present a message or a symbol to the user, it must be understandable for him ✓

As you can see in the activity diagram the user can follow a precisely defined process in order to generate its FMU component. The generation steps are delimited by a dashed outline where the user will choose the programming language used for his simulation then enter the number of

variables (Inputs and outputs of his program) in order to dynamically generate an adapted interface so he can define names and types of the variables and enter the path of the project, then enter a name of a class to be generated and specify the path in which it will be located, this last will be generated programmatically then post-compiled [Zukowski, 2006] using the prepare-package phase of Maven[46] to make sure the resources will be correctly packaged, so the user can easily generate JAR (Java ARchive)[47] and use the FMI builder to generate the FMU component as shown in 49.

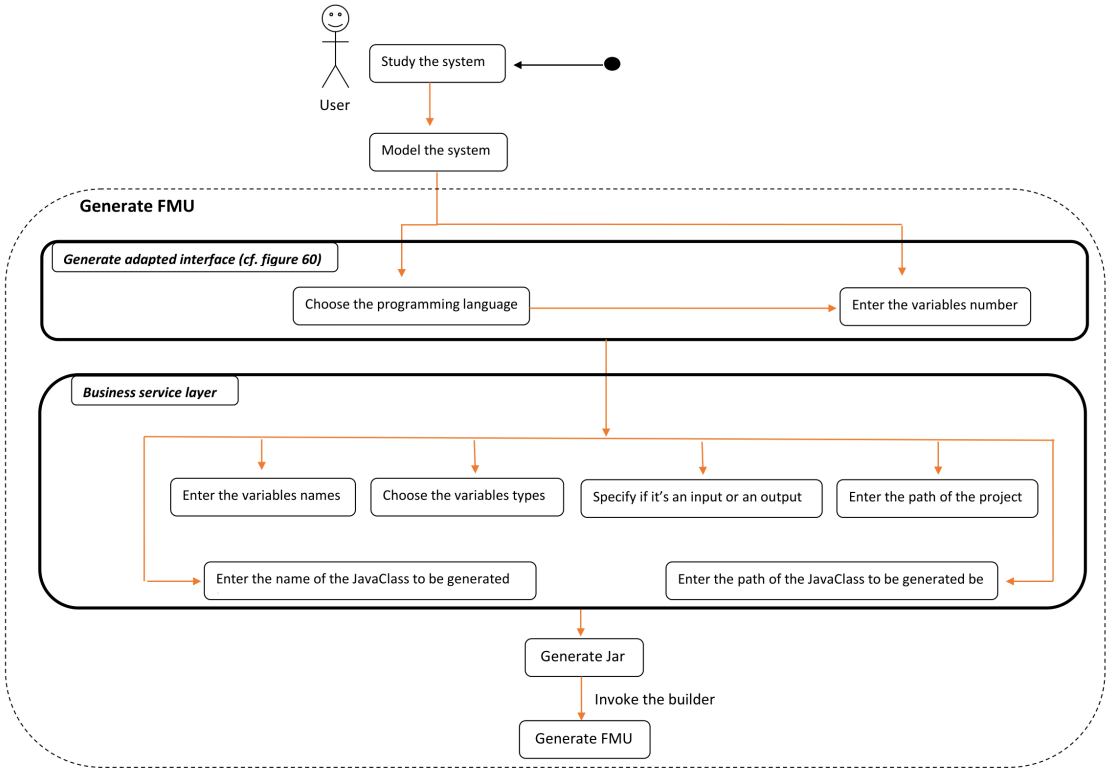

Figure 49: activity diagram for FMU generation

1.4 User interface - FMU

Many efforts have been made aiming to implement and test the FMI standard since its release. Technical issues and implementation of a generic interface to support the import of FMUs into a simulator is discussed in [Chen et al., 2011]. [Noll et al., 2011] describes the implementation of FMI in SimulationX. In [Elsheikh et al., 2013] an integration strategy for rapid prototyping for Modelica models into the FMI standard is presented. Since the modelling and simulation step is done separately by each designer, where the model has to be set up and tested against its specifications first. Standalone executables may be executed, traced, and debugged using additional

[46] build automation tool used primarily for Java projects

[47] package file format aggregating many Java class files and associated metadata and resources (text, images, project.) into one file for distribution

tools like an integrated development environment. The next step in the FMU generation process is to determine the interface of the simulation model, which is later exposed through the FMI. This consists of the definition of input and output quantities, or states, as well as internal timing (accuracy and precision required by the simulation model) and external timing (simulation step size for data exchange considerations. These informations are gathered with the information about the FMUs architecture in a modelDescription.xml file, which is connected to the software code using functions usually provided by the FMU SDK (software development kit) [Widl et al., 2013]. Another challenge is to fill-up the gap between the semantics of FMI and the semantics of the source formalism of the various calculation models (state machines, discrete event, data flow or timed automata) [Tripakis, 2015].

While FMU 3.1.5 parts are compiled and assembled. We get a zip file with a predefined structure. The .dll file is placed in the binaries folder for the corresponding platform. The modelDescription.xml file is placed in the top level (root) folder. The model source files are placed in the sources folder optionally.

We first analyzed the information about an FMU stored in the modelDescription.xml file. For example, the latter contains elements like **ModelVariables** defining all the variables of the FMU that are visible/accessible via the FMU functions. **ModelStructure** defining the structure of the model. Especially, the ordered lists of outputs, continuous-time states and initial unknowns which are mandatory and others like **VendorAnnotation** defining additional data that a vendor might want to store and that other vendors might ignore **TypeDefinitions** defining global list of type definitions that are utilized in ModelVariables, **LogCategories** defining the log information that is supported from the FMU, **DefaultExperiment** providing default settings for the integrator, such as stop time and relative tolerance which are optional and in our case which is FMI for co-simulation a **Cosimulation** element, describing if the slave can handle a variable communication step size or if it's able to interpolate continuous inputs, must be present. The modelDescription has attributes where the data are defined as the **fmiVersion**, **modelName**, guid to check that the XML file is compatible with the C functions of the FMU, the description, author and version of the model. The **generationTool**, **generationDateAndTime** and **variableNamingConvention** showing the convention followed while defining the variable names (flat or structured).

While generating the modelDescription file by the right data provided by the user. When using co-simulation standalone (fmi for co-simulation code generators are used to transfer models into compilable source code) instead of the co-simulation tool the user can wrap his model in order to generate the fmu file either by choosing to use JavaFMU adding fmu-framework-<version>.jar dependency to our model and extend FmiSimulation class then implement the methods and using fmu-builder-<version>.jar to generate the fmu from the jar. Or by including an fmu template header and manually transforming the model according to the FMI standard.

1.5 Scenario of tasks to generate an FMU from java program

In order to clarify the generation of FMUs for our designers, and to specify the requirements step C in (cf. Figure 47), we have detailed in simple words a scenario taking a java program (easy to use and simple to explain) as example. This generation task involves the following steps:
Open the IDE (eclipse)

Check that the app works before creating the artifact,
Create a main class for that (if it doesn't already exist) see (cf. Figure 50)

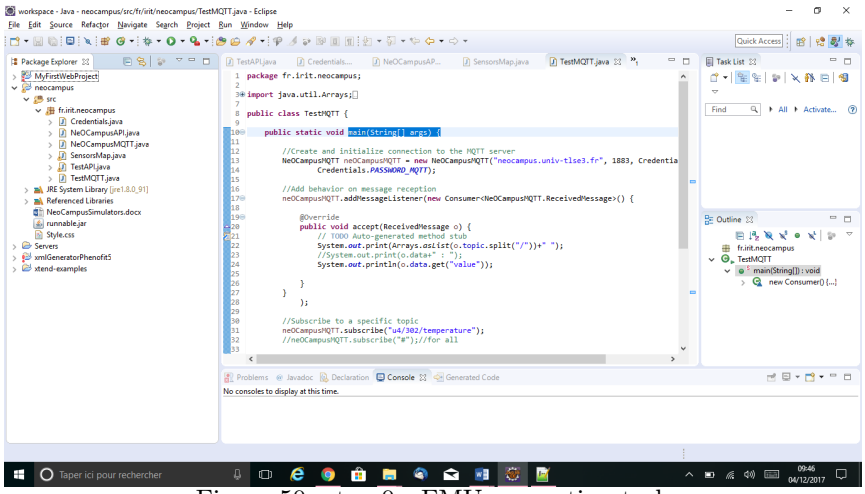

Figure 50: step 0 - FMU generation task

Run the java program and ensure that it works (right click on the main class, choose run as java application)see (cf. Figure 51)

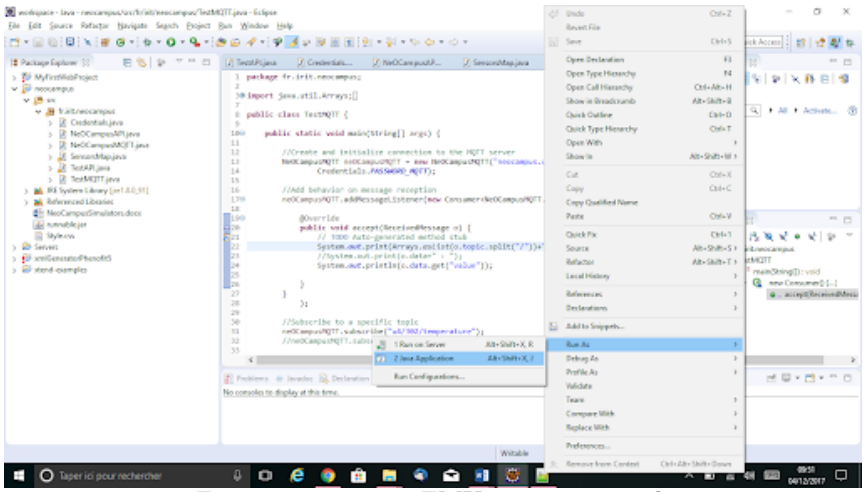

Figure 51: step 1 - FMU generation task

Click on the project name (right click) and choose the option "Build Path|Add External Archive" and chooses from the hard disk 3 files (.jar): "fmu_wrapper-2.24.0.jar, jna-platform-4.5.0.jar and simple-xml-2.7.1.jar" see (cf. Figure 52)

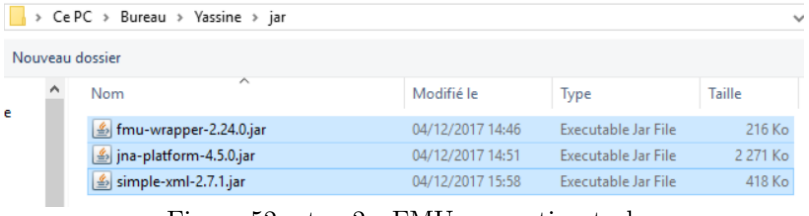

Figure 52: step 2 - FMU generation task

Add a file by clicking on the project name (right click) and choose the option "New | Package" and created the package "siani.javafmi" (cf. Figure 53)

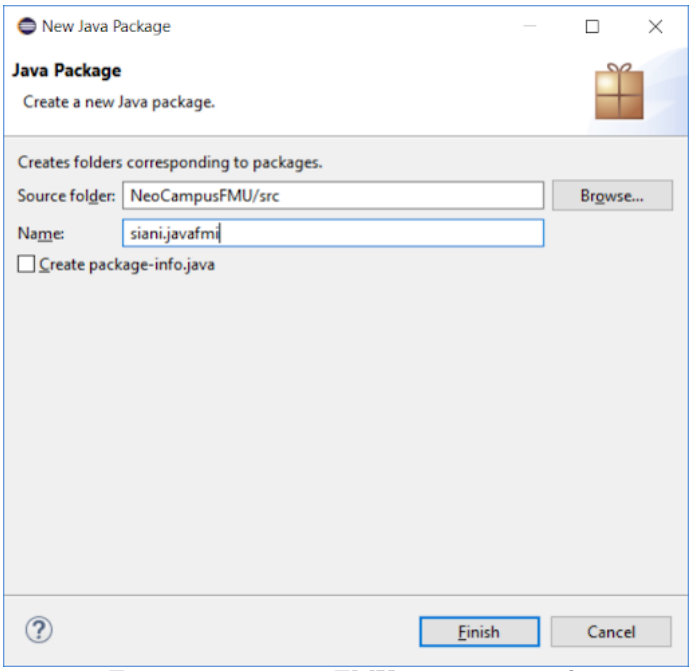

Figure 53: step 3 - FMU generation task

Click now on the package name (right click) and create a new class called "NCCosim" leaving the options unchanged (cf. Figure 54)

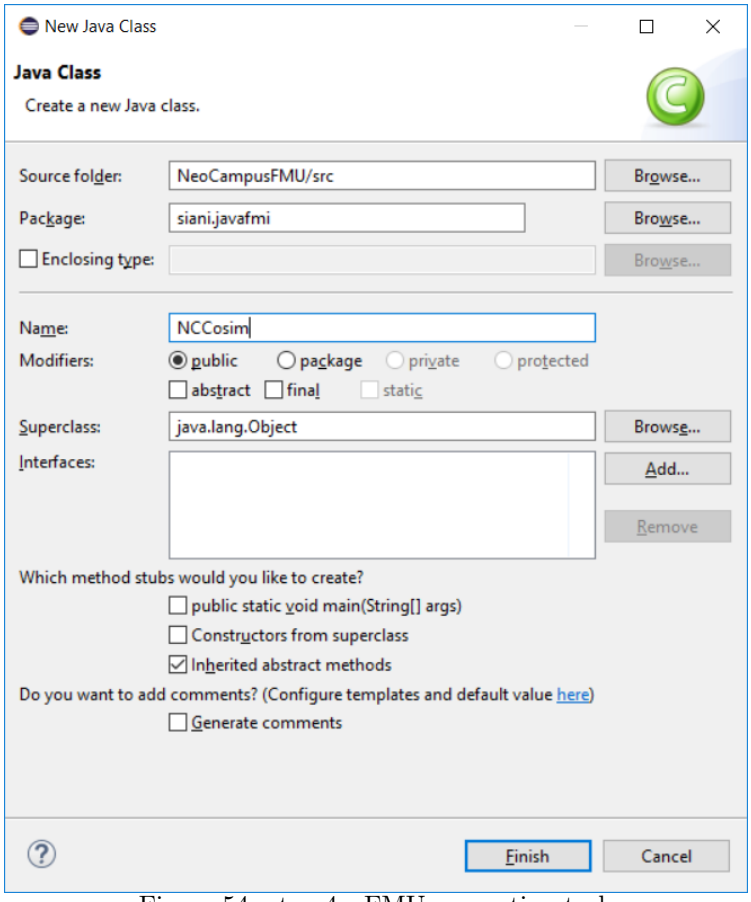

Figure 54: step 4 - FMU generation task

click now on the name of the file to type its code (cf. Figure 55)
In line 2, add the following references:
import org.javafmi.framework.FmiSimulation;
import org.javafmi.framework.State;

```
1  package siani.javafmi;
2
3  import org.javafmi.framework.FmiSimulation;
4  import org.javafmi.framework.State;
5
6  public class NCCosim {
7
```

Figure 55: step 5 - FMU generation task

Then modify line 6 by adding the words "extends FmiSimulation". An icon appears on line 6 (cf. Figure 56)

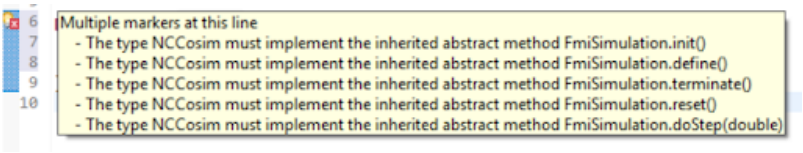

Figure 56: step 6 - FMU generation task

Clicking on the icon on the top left, click on the option "Add unimplemented methods" (cf. Figure 57)

```
public class NCCosim extends FmiSimulation {
    public NCCosim() {}

    @Override
    public Model define() {
        // TODO Auto-generated method stub
        return null;
    }

    @Override
    public Status doStep(double arg0) {
        // TODO Auto-generated method stub
        return null;
    }

    @Override
    public Status init() {
        // TODO Auto-generated method stub
        return null;
    }

    @Override
    public Status reset() {
        // TODO Auto-generated method stub
        return null;
    }

    @Override
    public Status terminate() {
        // TODO Auto-generated method stub
        return null;
    }
```

Figure 57: step 7 - FMU generation task

- public Status init(): in order to handle the values to read or write from an FMU wrapper (i.e to

read a variable with the name "port" and type "integer" use the syntax "registerInteger(port, () -> port, value -> port = value);" and to write on a variable with the name "temperature" and type "string" use the syntax "registerString(temperature, () -> temperature);")

- public Model define(): in order to define the model description use the syntax "return model(ModelName). canGetAndSetFMUstate(true) .add(variable(*temperature*).*asString()*.causality(*output*).variability(*discrete*));" add as much variables as you have specifying the name (temperature, port), the type (asString, asInteger, etc.), the causality (input,output), the variablity (discrete, ...)

- public Status doStep(double stepSize): in order to do an animation step; the simulation of a slave simulator within a communication interval is performed (i.e returning the calculated temperature for example, it advances the state and local time of the FMU by "stepSize" time units)

- public Status reset(): in order to reset the FMU back to its original state. Note that the environment has to initialize the FMU again after this function-call

- public Status terminate(): in order to terminate the simulation of the FMU

- Optional methods getState() and setState(): FMUs can save their state in order to use it to roll-back the simulation later on. By default the state saves all variables that have been registered

Run the project after all these modifications as we did in 1.5, if it works export the project as jar (right-click on the project then choose "export|runnable jar file", then modify the name of the exported jar as the name of the class extending FmiSimulation (cf. Figure 58)

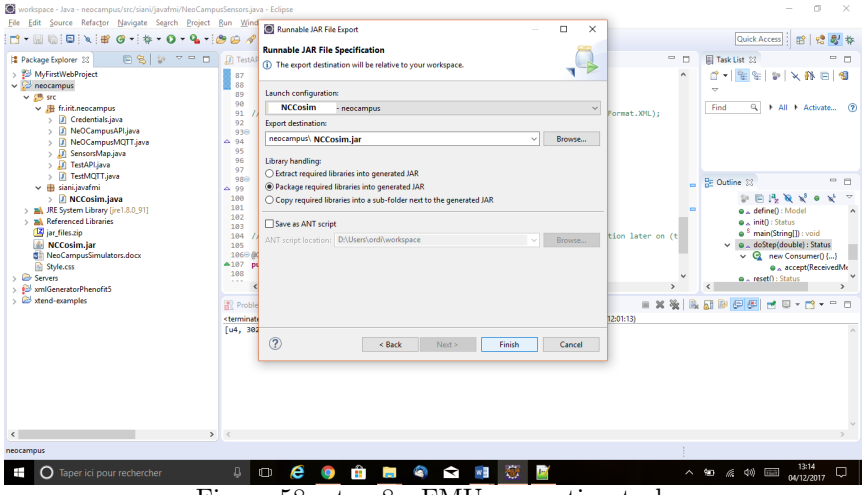

Figure 58: step 8 - FMU generation task

Check your artifact works: once the artifact generated, it should be able to run it outside of the development environment (eclipse). "java -jar NCCosim.jar" should work properly. If not, it won't work inside the fmu.
Invoke the builder to generate an FMU from a .jar; Go to the folder where the NCCosim.jar is and click both "Maj" and "right-click" to open the command line application, then use the command "java -jar fmu-builder-2.4.4.jar NCCosim.jar"

This inventory made it possible to extract the minimum necessary and sufficient variables for a simulation in order to generate as easily as possible the FMU component allowing co-simulation between systems. Then, we built a task model to understand the steps to be performed, in order to build an FMU component, and proposed several models (cf. Figure 59 and cf. Figure 60).

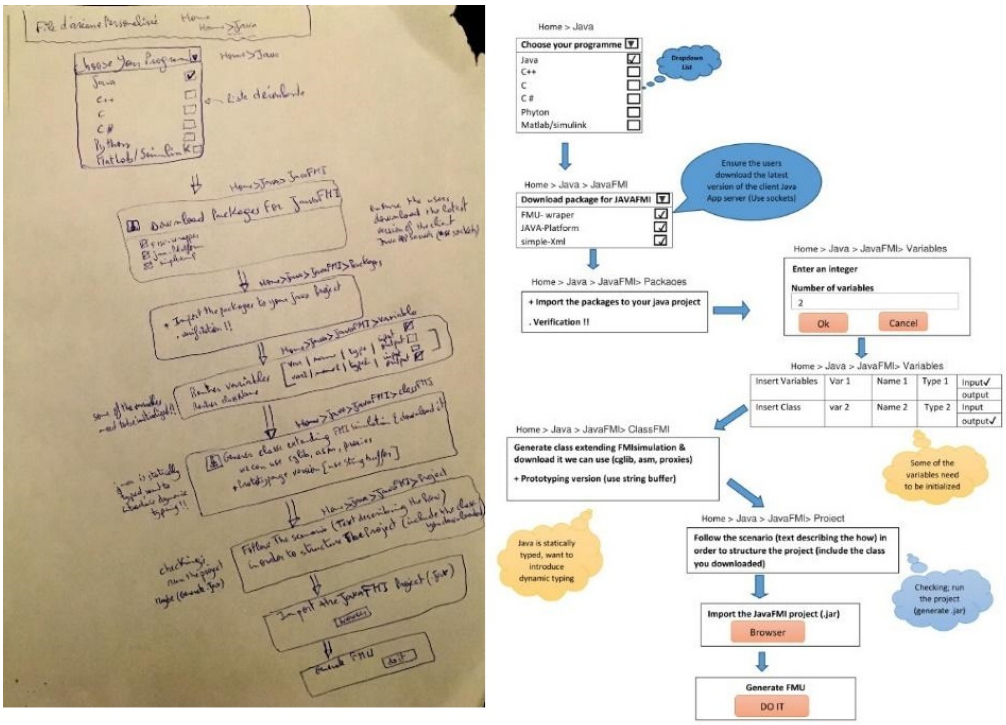

Figure 59: Low-fidelity models implementing the scenario

In the first low fidelity model (cf. Figure 59) we explained the scenario starting from the drop-down list proposed for users to choose the language used in their program, and for the java case which was detailed see (cf. Figure 60) to be implemented in our medium fidelity model see (cf. Figure 62), choose the libraries to download and check the version. Then we ask them to enter the number of variables they want to exchange with the other components (the required and provided interfaces). Then we ask them to enter the name of each variable (with a spellchecker) its type (to be

chosen) and if it's an input or output variable. Then with these data we generate a class extending FMISimulation class provided with the JavaFMI Package (we had doubt concerning this phase and if we give a scenario to follow in order to generate the class and structure the project or we automatize it; if we do not lose in the learning process). Then the user can generate a '.jar' project in order to generate from it a functional mock-up unit '.fmu'. This prototype helped understand what we should implement (automatable or not) and how we will do it in each phase of the process. We realized that we need also a text-box to describe and accompany the user in each phase of the process.

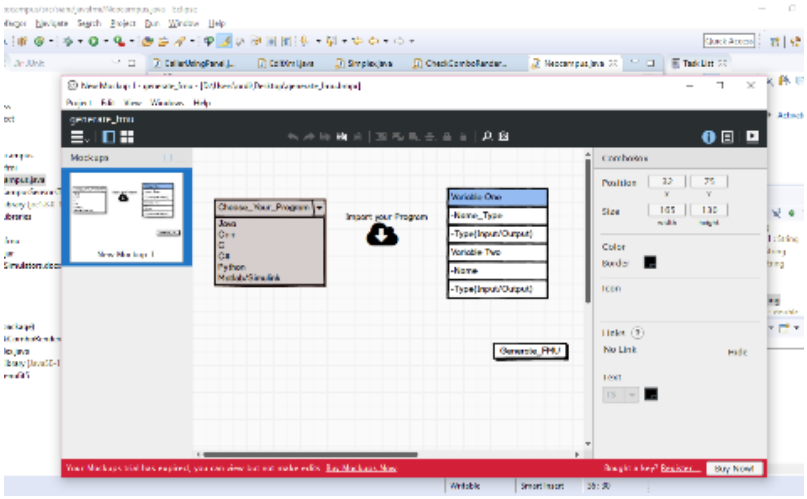

Figure 60: Low-fidelity model resulting from task analysis

In (cf. Figure 59) we describes in details the steps leading to generate the FMU components.

These steps were essential to identify the automatable steps of those where the actions of the user are essential.
Finally, based on these identified tasks, we proposed a "medium fidelity" interface see (cf. Figure 61), functional on an identified scenario for which we conducted a pre-experiment. In this figure we have an overview of the generation process where we go from a simulation knowing its input and output variables and generate a black box FMU component ready to be plugged into our co-simulation platform.

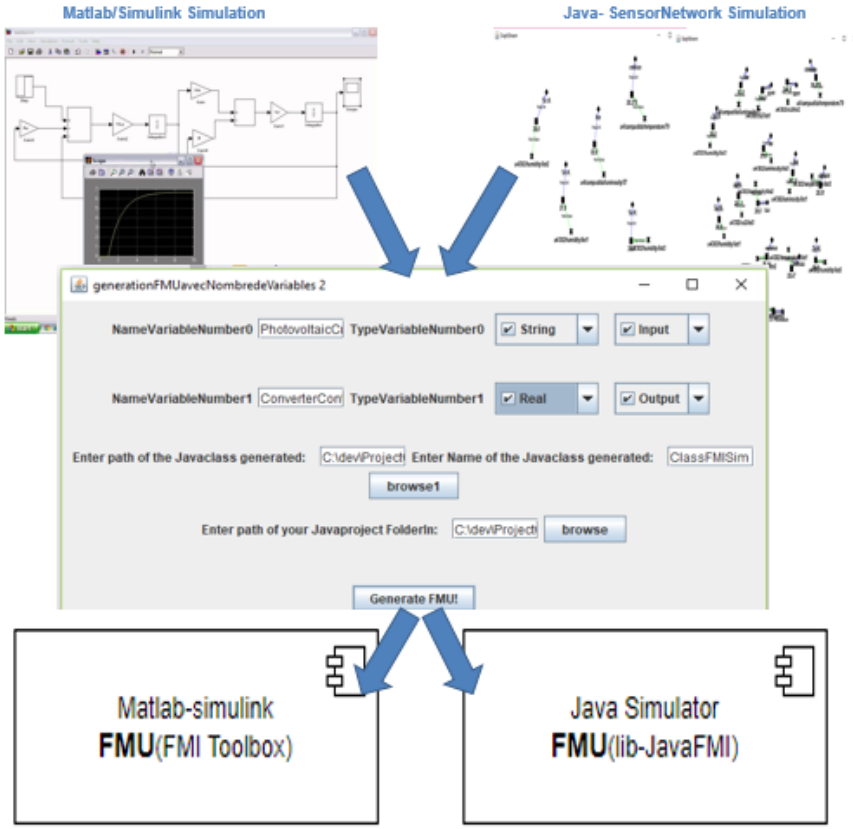

Figure 61: FMI component generation process

1.6 Pre-experiment

As mentioned earlier in 1, we found in surveys that a large majority of users were unfamiliar with FMI technology and had trouble "to jump in". That's why we wanted to check that using an interface, to help with the elaboration of FMU components, made sense for the designers (to understand the process and allow to collaborate) and that it was useful (in terms of time saving in the construction of a component for example, ...).

1.7 Participants

We conducted this pre-experiment with 7 novice participants and FMI experts. 6 participants were adult men (mean age = 20 years old, standard deviation = 5) and 1 woman (age = 27 years old). Participants were recruited from xxxx, xxxx and xxxx laboratories and were familiar with computers.

1.8 Equipment

We have developed the prototype under Java/QT on a laptop running Windows 10 OS (Core I7, 32 GB RAM, 17" screen).

1.9 Procedure

After signing the consent form, we exposed the two tasks to be performed for the pre-experimentation.
Task (1): an open program in the Eclipse framework was provided to the participants. The program was the same for all participants. They were asked to generate an FMU component using the latest version of the JavaFMI library and following the script available via a tutorial we provided them with (library download, class creation, component generation fmu2).
For the second **task (2)**, we asked them to launch and use the graphical interface, helping with the FMU component's generation, that we developed. This interface proposes to guide the user in the generation process from the number of input and output variables, entering names and types of variables, respect of cast and spellchecker, until the download of the generated component, providing a summary of the performed tasks (cf. Figure 62).

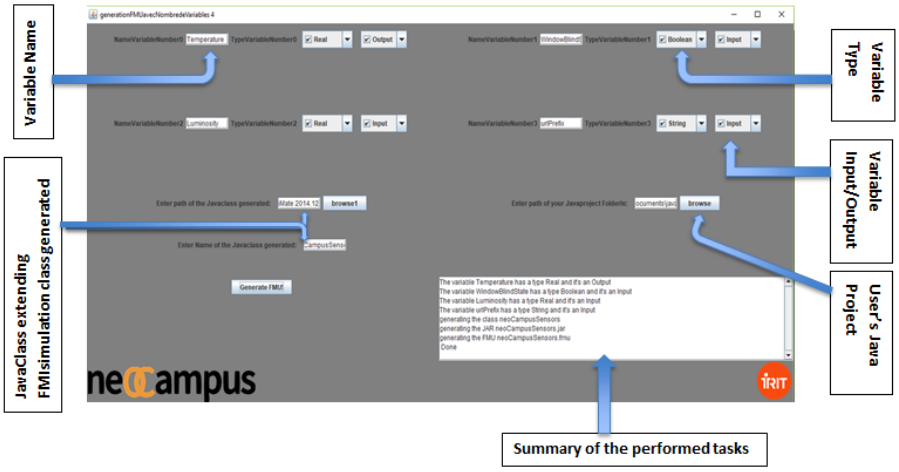

Figure 62: Screen-shot of the proposed interface

Six participants (number P1 à P5) performed the task (1) then (2) and the last two ones (number P6 et P7) performed the task (2) before (1) to counterbalance learning effects.

1.10 Analysis

We recorded and analyzed the processing time of the different participants either by following the prescribed scenario or by using our interface. The number of actions performed was also recorded and compared to the fact that the participants were FMI experts or not, or if they used to work

with integrated development environments (and more specifically Eclipse).

- **Independent Variables:** with interface, without interface.
- **Dependent Variables:** time (continuous variable), number of actions (discrete variable), FMI expertise: expert or not (categorical variable), experience related to frequency of use of IDE (categorical variable).

The aim is both to verify our initial hypotheses (the main objective of the tool is to allow non-experts to generate FMU components for co-simulation) and to iterate on our solution. We completed the pre-experimentation phase by debriefing with the participants.

2 Preliminary results and discussion

Users gave very mixed opinions on the experience, but most of them told us that following the script and trying to complete the program, to compile it and run it was a daunting task. In fact, one of these participants decided to leave the experiment after 32 minutes and found that the exercise was difficult.

6 users could reach the end of the task (1) but actually only 3 were able to generate a FMU component (two of which were experts FMI and one who frequently used the IDE and java as the main language in everyday work). The other 3 committed code errors that blocked this generation. On the other hand, the generation of the FMU component using our interface (task 2) was successful by all the participants.
We were able to realize that the order of the tasks did not matter much regarding the results in terms of time of completion of the task.

Participant	Time (1)	Time (2)	IDEuses	FMI expert
P1	25’	3’	4	oui
P2	32’ -ab.	5’	4	non
P3	42’	7’	3	non
P4	65’	14’	1	non
P5	54’	8’	2	non
P6	34’	10’	3	non
P7	30’	13’	4	oui

Figure 63: Results – task completion time

(cf. Figure 63) describes the times performed by the 7 participants (participants P6 and P7 first performed task (2) before (1)).
Participants are distinguished according to their level of FMI expertise and their use of IDEs (with scale: 1: never, 2: rarely, 3: regularly, 4: all the time).
There is systematically (see (cf. Figure 64)) a longer realization time (at least a factor of 3) for the

execution of the task (1) than for the task (2) regardless of their expertise. The ANOVA analysis also reveals a significant effect of the experience of an IDE on the time of completion of the task ($F(1,8)=50.02$, $p < 0.001$).

Figure 64: Task completion time for tasks (1) and (2)

With only two expert users, we cannot say much about the impact of the expertise on the time of completion of the task. Nevertheless, we can note here again a saving of time of realization by using our interface, independently of the degree of expertise of the subject. The number of actions performed (being analyzed) is also sharply down (by a factor of 8).

3 Conclusion and discussion

This preliminary work enabled us to first make an inventory of the practices of the different actors of the neOCampus project. In order to allow the different experts to communicate and collaborate, we understood that it was preferable that they could first and foremost keep their own practices by allowing them to build or improve their own "business" simulator.

We therefore opted for the interoperability of these heterogeneous simulators based on the FMI co-simulation standard overcoming model semantic gaps and offering them a validation platform for co-simulation. In this process, we have targeted one of the most difficult tasks, which is the generation of pluggable components in our platform, from heterogeneous simulations. We have therefore designed an interface facilitating this generation by focusing on automation with an understanding of the generation process. Hence, we pre-evaluated our interface in order to improve it. Our interface will aim the integration of the total control of the co-simulation with potentially visualization tools adapted to each participant in this co-simulation according to its needs.

The experience was enriching and appreciated. This constantly improving interface will not only be a mechanism to facilitate collaboration and simplify the co-simulation of our different systems, but can also be a plus for the FMI community that in all areas and for different uses is brought

to generate components and for some languages and simulation environment for which there is currently no documentation or library to do so. The UX (User eXperience) design focuses on making our application: more useful, more usable, more desirable, more navigable, more accessible, more credible.

As we interact with the different designers, had discussions with them and made them participate to the co-simulation process we noticed that in this human computer interaction field and precisely in visualization more work could be done especially while the co-simulaion in running, designers would appreciate having a clear vision of their system interacting with the whole. The biggest challenge would be to translate big data sets into interactive meaningful visualizations that meet the specific user needs (when to use which types of visualizations). Intuitive interaction and simple design that is understandable for non-informatics and non-statistic experts are crucial. Also with creating interactive visualizations for example, that will allow you to change various parameters so the user can play with them to see different perspectives of the same data over varying levels of time, data size, etc. To remove meaningless information (i.e. noise). When presented to the user in a defined context the first impression of a visualization should trigger intended actions.

CHAPTER VI: Conclusion and future work

conclusion It is common accepted that complex systems or cyberphysical systems need co-simulation for their study. Further more, they are made of heterogeneous sub-systems that have to exchange data. Usually each sub-system is modeled using specific tools, environments and simulators. The simulators have to interoperate to realize all the simulation of the system. It is known that interoperativity is a broad and complex subject. Interoperability is a strong commitment as the communication solution in heterogeneous systems.

We detailed in this work the different interoperability solutions (data, models and computation codes), and our interoperability approach (two levels: Structural and Semantic) was described and justified. For that, in the structural level, a standards interfaces table was constructed and the choice was made on FMI (Functionnal mockup interface), and more specifically FMI for co-simulation, which has been adapted to our needs. knowing that the goal was also to achieve the semantic interoperability level.
In our first contribution we described a co-simulation framework interoperability based FMI (Functional Mock up Interface) standard for the structural part and data mediation for semantic part. We present a case study for neOCampus project that shows how the framework helps to build the semantic interoperability of a cyberphysical system.

We have implemented our architecture and our modeling works well, as it was validated by domain experts. We took as example the 4 simulators including Contiki which is not compatible with Cosimate and for which we had to generate a slave FMU. Our database was encapsulated using JFMI. We were able to solve not only these structural problems but we added mediators to our platform in order to achieve semantic interoperability. It is necessary to mention that our framework allows the integration of all types of simulators and that for the non FMI, and even if they are not supported by cosimate, the use of a wrapper is enough to envelop them with a c code in order to connect them to the cosimate bus. This work allowed us to first make an inventory of the practices of the various actors of the neOCampus project. In order to allow the various experts to communicate and collaborate. We realized that it was preferable for them to keep their own practices by allowing them to build or improve their own "expert" simulator. Thus, the objective is a completely open system, easy to use, accepting all types of simulators.
In our second contribution, we propose a *dynamic* mediation for adaptation of the data which allows to go further towards the development of global and open simulation environment. We noticed that the semantic interoperability based on mediation was integrated to our framework in an ad-hoc way. This took a heavy work investment as each component needed a hand-made encapsulated mediator. The idea is to take advantage of the self-adaptation and openness capability of Adaptive Multi-Agent Systems (AMAS) through DREAM Belghache et al. [2017] and AMOEBA Verstaevel et al. [2017] (see section 2) which need no knowledge about the data and their application field. Bringing some dynamicity to this semantic layer.
As in our first contribution Motie et al. [2017] we designed a co-simulation framework interoperability performing a **co-simulation** based on a black box **component approach**. We defined communication ports and guaranteed the possibility of connection by respecting data types and the direction of the ports following the **FMI standard**. We designed a **mediation approach** to ensure the unambiguous information exchange.

As alternative we focus on Distributed Artificial Intelligence (DAI) which can be used in all the Artificial Intelligence (AI) problem domains. More precisely we focus in this "DAI subset" on Multi-agent systems (MAS) where collective behaviors emerge from the interaction of decentralized self-organized agents, and which was adapted to respond to our need. For instance it can also include machine learning (an agent could be a machine learning algorithm).

Hence the idea to take advantage of the self-adaptation and openness capability of Adaptive Multi-Agent Systems (AMAS) and to automate this mediation process using *DreAMoeba*, the extension of DREAM with AMOEBA which have been described in detail in the 3rd chapter, as a black box component in the co-simulation framework, which is able to link dynamically correlated inputs and outputs and hence to continuously adapt the structural interoperability and carry out the semantical one. Our DreAMoeba component which is used to ensure the mediation, in order to achieve the semantic interoperability, was encapsulated using JavaFMI.
DREAM, as a black box, offers openness and self-healing features (make new connections and replace lost connections on the fly) to our co-simulation framework when combined with an self-adaptive data translator will provide autonomous semantic interoperability.

We assumed that the AMAS used in the semantical interoperability level of our work could be less expressive but don't need any expert knowledge which is one of the strong point that leaded us to use them in addition to the fact that they are autonomous and adaptable.
The similarity of AMAS with the world of complex systems (AMAS is an example that has been proven) strengthens our choice. Despite the fact that AMAS doesn't give perfect results, but rather good, they work in any case, whatever the possible evolution of the system is, AMAS will always be able to respond.

limitations and prospects Furthermore, these new features, without taking into account the semantic aspect through the data translation, can be applied independently to improve the current tools relying on FMI by (semi-)automatize the linking process between the slaves inputs and outputs. Moreover we would like, as future work, to approach semantic interoperability using ontology for the comparison purposes.

Another problem identified, increasing the lack of communication and collaboration between the different designers, concerns the difficulty of being able to generate the co-simulation components (essentially components implementing the FMI) from the different simulators used. Indeed, the majority of the designers are experts in their field but 1/ are not necessarily computer experts and 2/ if they are, not often expert in co-simulation nor in FMI.
Their models were built completely in a disconnected way and do not benefit from the exchange of information that can simplify and accelerate their work.
This time of taking over the FMI technology (time and practice) has led us to propose, in our third contribution, a mechanism to support the generation of "black-box" FMU components based on user practices. The transition from the subsystem studied to an FMU component (black box) requires knowledge of the FMI standard, thus constituting an obstacle for our designers, inhibiting thereby their collaborations. This interface is intended to be easy to use for novice users, adapted and allows to accompany the process of generation of components ready to connect to our platform.
This assistance tool to design interoperable components will encourage the future collaborators participating in the neOCampus operation to communicate with the other subsystems improving

the quantitative evaluation of the co-simulation engine, and more specifically the evaluation of the accuracy of the prediction (e.g., with cross-validation), how it scales with increasing data, the time required to compute a solution with different training data size. This should enrich our framework, further improve the accuracy of the results provided, and facilitate the exchange and analysis in our complex system. It will also be useful for the FMI community usually brought to generate FMU components, knowing that for some languages and simulation environment there is currently no documentation or library to do so.

Annexes

References

[1] UCI Machine Learning Repository: Data Sets, . URL `https://archive.ics.uci.edu/ml/datasets.html?format=&task=&att=&area=&numAtt=&numIns=&type=ts&sort=nameUp&view=list`.

[2] UCI Machine Learning Repository: Ozone Level Detection Data Set, . URL `https://archive.ics.uci.edu/ml/datasets/Ozone+Level+Detection`.

[3] Iso/Tr 16982. Ergonomics of human-system interaction–usability methods supporting human-centred design, 2002.

[4] Chadia Abras, Diane Maloney-Krichmar, and Jenny Preece. User-centered design. *Bainbridge, W. Encyclopedia of Human-Computer Interaction. Thousand Oaks: Sage Publications*, 37(4): 445–456, 2004.

[5] Hannele Ahvenniemi, Aapo Huovila, Isabel Pinto-Seppä, and Miimu Airaksinen. What are the differences between sustainable and smart cities? *Cities*, 60:234–245, 2017.

[6] Gilles Akoun and J-P Yonnet. 3d analytical calculation of the forces exerted between two cuboidal magnets. *IEEE Transactions on magnetics*, 20(5):1962–1964, 1984.

[7] Ahmad T Al-Hammouri. A comprehensive co-simulation platform for cyber-physical systems. *Computer Communications*, 36(1):8–19, 2012.

[8] Eiman Al Nuaimi, Hind Al Neyadi, Nader Mohamed, and Jameela Al-Jaroodi. Applications of big data to smart cities. *Journal of Internet Services and Applications*, 6(1):1–15, 2015.

[9] Vito Albino, Umberto Berardi, and Rosa Maria Dangelico. Smart cities: Definitions, dimensions, performance, and initiatives. *Journal of Urban Technology*, 22(1):3–21, 2015.

[10] Sebastien Allain, Jean-Philippe Chateau, and Olivier Bouaziz. Constitutive model of the twip effect in a polycrystalline high manganese content austenitic steel. *steel research international*, 73(6-7):299–302, 2002.

[11] Grégoire Allaire. *Analyse numérique et optimisation: une introduction à la modélisation mathématique et à la simulation numérique*. Editions Ecole Polytechnique, 2005.

[12] Pierre America. Designing an object-oriented programming language with behavioural subtyping. In *Workshop/School/Symposium of the REX Project (Research and Education in Concurrent Systems)*, pages 60–90. Springer, 1990.

[13] Rob Armstrong, Dennis Gannon, Al Geist, Katarzyna Keahey, Scott Kohn, Lois McInnes, Steve Parker, and Brent Smolinski. Toward a common component architecture for high-performance scientific computing. In *High Performance Distributed Computing, 1999. Proceedings. The Eighth International Symposium on*, pages 115–124. IEEE, 1999.

[14] Naveen Ashish and Jose-Luis Ambite. *Data Integration in the Life Sciences: 11th International Conference, DILS 2015, Los Angeles, CA, USA, July 9-10, 2015, Proceedings*, volume 9162. Springer, 2015.

[15] Yacine Atif, Sujith Samuel Mathew, and Abderahmane Lakas. Building a smart campus to support ubiquitous learning. *Journal of Ambient Intelligence and Humanized Computing*, 6 (2):223–238, 2015.

[16] R. Axelrod. *The Evolution of Cooperation*. Basic Books, 1984.

[17] AV Barzykin, K Seki, and F Shibata. Periodically driven linear system with multiplicative colored noise. *Physical Review E*, 57(6):6555, 1998.

[18] JM Christian Bastien and Dominique L Scapin. *Ergonomic criteria for the evaluation of human-computer interfaces*. PhD thesis, Inria, 1993.

[19] Oliver Emile Glaves Bates and Adrian John Friday. Beyond data in the smart city: learning from a case study of re-purposing existing campus iot. *IEEE Pervasive Computing*, 16(2): 54–60, 2017.

[20] Vladimir Bazjanac and DB Crawley. Industry foundation classes and interoperable commercial software in support of design of energy-efficient buildings. In *Proceedings of Building Simulation'99*, volume 2, pages 661–667, 1999.

[21] A Beaudre and A Pica. Simulation virtuelle: moyens et méthodologie. *Cancer/Radiothérapie*, 1(5):573–580, 1997.

[22] Elhadi Belghache, Jean-Pierre Georgé, and Marie-Pierre Gleizes. DREAM: Dynamic data Relation Extraction using Adaptive Multi-agent systems (regular paper). In *IEEE International Conference on Digital Information Management (ICDIM)*. IEEE, 2017.

[23] Ron Ben-Natan. *Corba: a guide to common object request broker architecture*. McGraw-Hill, Inc., 1995.

[24] Marco Bernardo, Paolo Ciancarini, and Lorenzo Donatiello. Architecting families of software systems with process algebras. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 11(4):386–426, 2002.

[25] Olivier Bertrand, Patrice Carle, and Christine Choppy. Modelling chronicle recognition for distributed simulation processing with coloured petri nets. In *Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, page 42. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2009.

[26] Jean Bézivin and Jean-Pierre Briot. Sur les principes de base de l'ingénierie des modèles. *L'OBJET*, 10(4):145–157, 2004.

[27] Vasudha Bhatnagar and Srinath Srinivasa. *Big Data Analytics: Second International Conference, BDA 2013, Mysore, India, December 16-18, 2013, Proceedings*, volume 8302. Springer, 2013.

[28] Benjamin S Blanchard. *System engineering management*. John Wiley & Sons, 2004.

[29] Torsten Blochwitz, Martin Otter, Martin Arnold, Constanze Bausch, H Elmqvist, A Junghanns, J Mauß, M Monteiro, T Neidhold, Dietmar Neumerkel, et al. The functional mockup interface for tool independent exchange of simulation models. In *Proceedings of the 8th International Modelica Conference; March 20th-22nd; Technical Univeristy; Dresden; Germany*, number 063, pages 105–114. Linköping University Electronic Press, 2011.

[30] Torsten Blochwitz, Martin Otter, Johan Akesson, Martin Arnold, Christoph Clauss, Hilding Elmqvist, Markus Friedrich, Andreas Junghanns, Jakob Mauss, Dietmar Neumerkel, et al. Functional mockup interface 2.0: The standard for tool independent exchange of simulation models. In *Proceedings of the 9th International MODELICA Conference; September 3-5; 2012; Munich; Germany*, number 076, pages 173–184. Linköping University Electronic Press, 2012.

[31] Csaba Attila Boer and Alexander Verbraeck. Distributed simulation and manufacturing: distributed simulation with cots simulation packages. In *Proceedings of the 35th conference on Winter simulation: driving innovation*, pages 829–837. Winter Simulation Conference, 2003.

[32] Jérémy Boes, Frédéric Migeon, Pierre Glize, and Erwan Salvy. Model-free optimization of an engine control unit thanks to self-adaptive multi-agent systems. In *International Conference on Embedded Real Time Software and Systems (ERTS2), Toulouse*. SIA/3AF/SEE, 2014.

[33] Jérémy Boes, Julien Nigon, Nicolas Verstaevel, Marie-Pierre Gleizes, and Frédéric Migeon. The self-adaptive context learning pattern: Overview and proposal (regular paper). In *International and Interdisciplinary Conference on Modeling and Using Context (CONTEXT), Larnaca, Cyprus,*, number 9405 in LNAI, pages 91–104. Springer, 2015.

[34] Guy A Boy. The group elicitation method for participatory design and usability testing. *interactions*, 4(2):27–33, 1997.

[35] Tim Bray, Jean Paoli, C Michael Sperberg-McQueen, Eve Maler, and François Yergeau. Extensible markup language (xml). *World Wide Web Journal*, 2(4):27–66, 1997.

[36] S Brueckner, G. Di Marzo Serugendo, A. Karageorgos, and R. Nagpal, editors. *Engineering Self-Organising Systems, Methodologies and Applications*, volume 3464 of *LNAI*. Springer, 2005. ISBN 3-540-26180-X.

[37] S. Brueckner, G. Di Marzo Serugendo, D. Hales, and F. Zambonelli, editors. *Engineering Self-Organising Systems*, volume 3910 of *LNAI*. Springer, 2006.

[38] Don Brutzman, Michael Zyda, J Mark Pullen, Katherine L Morse, Steven Fouskarinis, David Drake, Dennis Moen, Curt Blais, Andrzej Kapolka, and Don McGregor. Extensible modeling and simulation framework (xmsf) challenges for web-based modeling and simulation. 2002.

[39] JP Calvez, D Heller, and O Pasquier. System performance modeling and analysis with vhdl: Benefits and limitations. In *Proceedings of VHDL-Forum Europe Conference*, volume 157, page 158, 1995.

[40] James Calvin, Alan Dickens, Bob Gaines, Paul Metzger, Dale Miller, and Dan Owen. The simnet virtual world architecture. In *Virtual Reality Annual International Symposium*, pages 450–455. IEEE, 1993.

[41] Benjamin Camus, Julien Siebert, Christine Bourjot, and Vincent Chevrier. Mod\'elisation multi-niveaux dans aa4mm. *arXiv preprint arXiv:1210.5936*, 2012.

[42] Benjamin Camus, Christine Bourjot, and Vincent Chevrier. Considering a multi-level model as a society of interacting models: Application to a collective motion example. *Journal of Artificial Societies and Social Simulation*, 18(3):7, 2015.

[43] Longbing Cao, Ana LC Bazzan, Andreas L Symeonidis, Vladimir Gorodetsky, Gerhard Weiss, and S Yu Philip. *Agents and Data Mining Interaction: 7th International Workshop, ADMI 2011, Taipei, Taiwan, May 2-6, 2011, Revised Selected Papers*, volume 7103. Springer, 2011.

[44] Longbing Cao, Yifeng Zeng, Andreas L Symeonidis, Vladimir Gorodetsky, S Yu Philip, and Munindar P Singh. *Agents and Data Mining Interaction: 8th International Workshop, ADMI 2012, Valencia, Spain, June 4-5, 2012, Revised Selected Papers*, volume 7607. Springer, 2013.

[45] Yongcan Cao, Wenwu Yu, Wei Ren, and Guanrong Chen. An overview of recent progress in the study of distributed multi-agent coordination. *IEEE Transactions on Industrial informatics*, 9(1):427–438, 2013.

[46] D. Capera, J.-P. Georgé, M.-P. Gleizes, and P. Glize. The AMAS Theory for Complex Problem Solving Based on Self-organizing Cooperative Agents. In *1st International TAPOCS Workshop at IEEE 12th WETICE*, pages 383–388. IEEE, 2003.

[47] François E Cellier. General system problem solving paradigm for qualitative modeling. In *Qualitative simulation modeling and analysis*, pages 51–71. Springer, 1991.

[48] Vincent Chapurlat and Nicolas Daclin. System interoperability: definition and proposition of interface model in mbse context. *IFAC Proceedings Volumes*, 45(6):1523–1528, 2012.

[49] David Chen. Enterprise interoperability framework. In *EMOI-INTEROP*, 2006.

[50] Wuzhu Chen, Michaela Huhn, and Peter Fritzson. A generic fmu interface for modelica. In *Proceedings of the 4th International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools; Zurich; Switzerland; September 5; 2011*, number 056, pages 19–24. Linköping University Electronic Press, 2011.

[51] Samir Chouali, Maritta Heisel, and Jeanine Souquières. Proving component interoperability with b refinement. *Electronic Notes in Theoretical Computer Science*, 160:157–172, 2006.

[52] MODELISAR Consortium et al. Functional mock-up interface for co-simulation. *Accessed March*, 1:2013, 2010.

[53] B. E. Cooper. Correlation and Function Fitting. In *Statistics for Experimentalists*, pages 206–212. Elsevier, May 2014. ISBN 978-1-4832-8052-3.

[54] Peter A. Corning. The re-emergence of emergence, and the causal role of synergy in emergent evolution. *Synthese*, 185(2):295–317, March 2012. ISSN 0039-7857, 1573-0964. doi: 10.1007/s11229-010-9726-2. WOS:000303530600009.

[55] Laurent Crépin. *Couplage de modèles population et individu-centrés pour la simulation parallélisée des systèmes biologiques: application à la coagulation du sang*. PhD thesis, Université de Bretagne occidentale-Brest, 2013.

[56] Enrico De Angelis, Angelo Luigi Camillo Ciribini, Lavinia Chiara Tagliabue, and Michela Paneroni. The brescia smart campus demonstrator. renovation toward a zero energy classroom building. *Procedia engineering*, 118:735–743, 2015.

[57] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[58] Benoît Delinchant, Pierre-Yves Gibello, Franck Verdière, and Frédéric Wurtz. Distribution des services de calcul pour la conception et la gestion optimale des bâtiments: perspectives des approches composants déployés via le" cloud computing". In *XXXe Rencontres AUGC-IBPSA*, 2012.

[59] G. Di Marzo Serugendo, A. Karageorgos, O. F. Rana, and F. Zambonelli, editors. *Engineering Self-Organising Systems, Nature-Inspired Approaches to Software Engineering* , volume 2977 of *LNCS*. Springer, 2004. ISBN 3-540-21201-9.

[60] Giovanna Di Marzo Serugendo, Marie-Pierre Gleizes, and Anthony Karageorgos. *Self-organising Software: From Natural to Artificial Adaptation*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011.

[61] Saikou Y Diallo, Andreas Tolk, Jason Graff, and Anthony Barraco. Using the levels of conceptual interoperability model and model-based data engineering to develop a modular interoperability framework. In *Proceedings of the Winter Simulation Conference*, pages 2576–2586. Winter Simulation Conference, 2011.

[62] Didier Donsez. Objets, composants et services: intégration de propriétés non fonctionnelles. *Habilitation à Diriger des Recherches de l'Université Joseph Fourier, http://wwwadele. imag. fr/users/Didier. Donsez/pub/publi/hdr*, 2006.

[63] J. Dréo, A. Pétrowski, P. Siarry, and E. Taillard. *Metaheuristics for Hard Optimization*. Springer, 2006.

[64] Desmond F D'souza and Alan Cameron Wills. *Objects, components, and frameworks with UML: the catalysis approach*, volume 1. Addison-Wesley Reading, 1998.

[65] Adam Dunkels, Bjorn Gronvall, and Thiemo Voigt. Contiki-a lightweight and flexible operating system for tiny networked sensors. In *Local Computer Networks*, pages 455–462. IEEE, 2004.

[66] Chuck Eastman, Paul Teicholz, Rafael Sacks, and Kathleen Liston. *BIM handbook: A guide to building information modeling for owners, managers, designers, engineers and contractors*. John Wiley & Sons, 2011.

[67] Atiyah Elsheikh, Muhammed Usman Awais, Edmund Widl, and Peter Palensky. Modelica-enabled rapid prototyping of cyber-physical energy systems via the functional mockup interface. In *Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES), 2013 Workshop on*, pages 1–6. IEEE, 2013.

[68] Mica R Endsley. *Designing for situation awareness: An approach to user-centered design*. CRC press, 2016.

[69] Olaf Enge-Rosenblatt, Christoph Clauß, André Schneider, and Peter Schneider. Functional digital mock-up and the functional mock-up interface-two complementary approaches for a comprehensive investigation of heterogeneous systems. In *Proceedings of the 8th International Modelica Conference; March 20th-22nd; Technical Univeristy; Dresden; Germany*, number 063, pages 748–755. Linköping University Electronic Press, 2011.

[70] Emeka Eyisi, Jia Bai, Derek Riley, Jiannian Weng, Wei Yan, Yuan Xue, Xenofon Koutsoukos, and Janos Sztipanovits. Ncswt: An integrated modeling and simulation tool for networked control systems. *Simulation Modelling Practice and Theory*, 27:90–111, 2012.

[71] Wei Fan and Albert Bifet. Mining big data: current status, and forecast to the future. *ACM sIGKDD Explorations Newsletter*, 14(2):1–5, 2013.

[72] J. Ferber. *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*. Addison Wesley, 1999.

[73] Tony Finch. Incremental calculation of weighted mean and variance. *University of Cambridge*, 4:11–5, 2009.

[74] PG Foley, F Mamaghani, and PA Birkel. The synthetic environment data representation and interchange specification (sedris) development project. In *Proceedings of the interservice/industry training, simulation, and education conference (I/ITSEC)*, 1998.

[75] Jonathan Friedman and Jason Ghidella. Using model-based design for automotive systems engineering-requirements analysis of the power window example. Technical report, SAE Technical Paper, 2006.

[76] Sana Gaaloul. *Interopérabilité basée sur les standards Modelica et composant logiciel pour la simulation énergétique des systèmes de bâtiment*. PhD thesis, Université de Grenoble, 2012.

[77] Virginie Galtier, Stephane Vialle, Cherifa Dad, Jean-Philippe Tavella, Jean-Philippe Lam-Yee-Mui, and Gilles Plessis. Fmi-based distributed multi-simulation with daccosim. In *Proceedings of the Symposium on Theory of Modeling & Simulation: DEVS Integrative M&S Symposium*, pages 39–46. Society for Computer Simulation International, 2015.

[78] Erich Gamma. *Design patterns: elements of reusable object-oriented software*. Pearson Education India, 1995.

[79] Jean-Marie Garcia, David Gauchard, Olivier Brun, Pierre Bacquet, James Sexton, and Eoin Lawless. Modélisation différentielle du trafic et simulation hybride distribuée. *Calculateurs parallèles*, 18(3), 2001.

[80] Salvador García, Julián Luengo, and Francisco Herrera. *Data preprocessing in data mining*. Springer, 2015.

[81] Alfredo Garro and Alberto Falcone. On the integration of hla and fmi for supporting interoperability and reusability in distributed simulation. In *Proceedings of the Symposium on Theory of Modeling & Simulation: DEVS Integrative M&S Symposium*, pages 9–16. Society for Computer Simulation International, 2015.

[82] Kyle Geiger. *inside ODBC*. Microsoft Press, 1995.

[83] J. P. Georgé, B. Edmonds, and P. Glize. *Making Self-organizing adaptive multi-agent systems work*, pages 321–340. Kluwer Publishing, 2004.

[84] Jean-Pierre Georgé, Bruce Edmonds, and Pierre Glize. Making self-organising adaptive multiagent systems work. In *Methodologies and software engineering for agent systems*, pages 321–340. Springer, 2004.

[85] M.-P. Gleizes, V. Camps, and P. Glize. A Theory of Emergent Computation Based on Cooperative Self-Oganization for Adaptive Artificial Systems. In 4^{th} *European Congress of Systems Science*, 1999.

[86] Marie-Pierre Gleizes, Valérie Camps, and Pierre Glize. A theory of emergent computation based on cooperative self-organization for adaptive artificial systems. In *Fourth European Congress of Systems Science*, pages 20–24, 1999.

[87] Marie-Pierre Gleizes, Jérémy Boes, Bérangère Lartigue, and François Thiébolt. neocampus: A demonstrator of connected, innovative, intelligent and sustainable campus. In *International Conference on Intelligent Interactive Multimedia Systems and Services*, pages 482–491. Springer, 2017.

[88] Cláudio Gomes, Casper Thule, David Broman, Peter Gorm Larsen, and Hans Vangheluwe. Co-simulation: State of the art. *arXiv preprint arXiv:1702.00686*, 2017.

[89] Ricardo Gomes, Henrique Pombeiro, Carlos Silva, Paulo Carreira, Miguel Carvalho, Gonçalo Almeida, Pedro Domingues, and Paulo Ferrão. Towards a smart campus: Building-user learning interaction for energy efficiency, the lisbon case study. In *Handbook of Theory and Practice of Sustainable Development in Higher Education*, pages 381–398. Springer, 2017.

[90] World Bank Group. *World development indicators 2014*. World Bank Publications, 2014.

[91] John Guckenheimer and Julio M Ottino. Foundations for complex systems research in the physical sciences and engineering. In *Report from an NSF workshop*, 2008.

[92] C Guglielmina and A Berre. Athena,"project a4"(slide presentation). *ATHENA Intermediate Audit, Athens*, 2005.

[93] G Guilizzoni. Balsamiq mockups, balsamiq, 2010.

[94] Valérian Guivarch, Valérie Camps, André Péninou, and Pierre Glize. Self-adaptation of a learnt behaviour by detecting and by managing user's implicit contradictions. In *IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT), Warsaw, Poland*. IEEE Computer Society, 2014.

[95] Felix Günther, Georg Mallebrein, and Heinz Ulbrich. A modular technique for automotive system simulation. In *Proceedings of the 9th International MODELICA Conference; September 3-5; 2012; Munich; Germany*, number 076, pages 589–598. Linköping University Electronic Press, 2012.

[96] Alon Y Halevy, Zachary G Ives, Dan Suciu, and Igor Tatarinov. Schema mediation in peer data management systems. In *Data Engineering, 2003. Proceedings. 19th International Conference on*, pages 505–516. IEEE, 2003.

[97] Jiawei Han, Micheline Kamber, and Jian Pei. *Data mining: concepts and techniques*. Elsevier, 2011.

[98] Jun Han. Temporal logic based specification of component interaction protocols. In *Proceedings of the Second Workshop on Object Interoperability ECOOP*, pages 12–16, 2000.

[99] Thomas Hansmann and Peter Niemeyer. Big data-characterizing an emerging research field using topic models. In *Web Intelligence (WI) and Intelligent Agent Technologies (IAT), 2014 IEEE/WIC/ACM International Joint Conferences on*, volume 1. IEEE, 2014.

[100] Colin Harrison and Ian Abbott Donnelly. A theory of smart cities. In *Proceedings of the 55th Annual Meeting of the ISSS-2011, Hull, UK*, volume 55, 2011.

[101] Sandra Heiler. Semantic interoperability. *ACM Computing Surveys (CSUR)*, 27(2):271–273, 1995.

[102] Jan Hensen, Ery Djunaedy, Marija Radošević, and Azzedine Yahiaoui. Building performance simulation for better design: some issues and solutions. In *Proceedings of the 21st Conference on Passive and Low Energy Architecture (PLEA)*, pages 1185–1190. Citeseer, 2004.

[103] Juan Hernández, José M Troya, and Antonio Vallecillo. Lesson 3: Component interoperability.

[104] Fabiano Hessel, P Le Marrec, Carlos A Valderrama, Mohamed Romdhani, and Ahmed Amine Jerraya. Mci—multilanguage distributed co-simulation tool. In *Distributed and Parallel Embedded Systems*, pages 191–200. Springer, 1999.

[105] F. Heylighen. Evolution, Selfishness and Cooperation; Selfish Memes and the Evolution of Cooperation. *Journal of Ideas*, 2(4):70–84, 1992.

[106] Gerhard Hippmann, Martin Arnold, and Marcus Schittenhelm. Efficient simulation of bush and roller chain drives. In *Proceedings of ECCOMAS Thematic Conference on Advances in Computational Multibody Dynamics, J. Goicolea, J. Cuadrado, and JG Orden, eds., Madrid, Spain, June*, pages 21–24, 2005.

[107] Ronghuai HUANG, Jinbao ZHANG, Yongbin HU, and Junfeng YANG. Smart campus: The developing trends of digital campus [j]. *Open Education Research*, 4(004), 2012.

[108] B.A. Huberman. *The performance of cooperative processes*. MIT Press / North-Holland, 1991.

[109] HV Jagadish, Johannes Gehrke, Alexandros Labrinidis, Yannis Papakonstantinou, Jignesh M Patel, Raghu Ramakrishnan, and Cyrus Shahabi. Big data and its technical challenges. *Communications of the ACM*, 57(7), 2014.

[110] Carlos A Jara, Francisco Esquembre, Francisco A Candelas, Fernando Torres, and Sebastián Dormido. New features of easy java simulations for 3d modeling. *IFAC Proceedings Volumes*, 42(24):250–255, 2010.

[111] DR Jefferson. Virtual time, acm transactions on programming language and systems, 1985.

[112] Maziar M Kandelous and Jiří Šimnek. Comparison of numerical, analytical, and empirical models to estimate wetting patterns for surface and subsurface drip irrigation. *Irrigation Science*, 28(5):435–444, 2010.

[113] James Max Kanter and Kalyan Veeramachaneni. Deep feature synthesis: Towards automating data science endeavors. In *Data Science and Advanced Analytics (DSAA), 2015. 36678 2015. IEEE International Conference on*, pages 1–10. IEEE, 2015.

[114] M Kasunic and W Anderson. Measuring systems interoperability: challenges and opportunities, software engineering measurement and analysis initiative. Technical report, Technical note CMU/SEI–2004–TN–003, 2004.

[115] John F Kelley. An iterative design methodology for user-friendly natural language office information applications. *ACM Transactions on Information Systems (TOIS)*, 2(1):26–41, 1984.

[116] Sameer Khader, Alan Hadad, and Akram A Abu-Aisheh. The application of psim & matlab/simulink in power electronics courses. In *Global Engineering Education Conference (EDUCON)*, pages 118–121. IEEE, 2011.

[117] Zaheer Khan, Ashiq Anjum, Kamran Soomro, and Muhammad Atif Tahir. Towards cloud based big data analytics for smart future cities. *Journal of Cloud Computing*, 4(1):1–11, 2015.

[118] Gregor Kiczales, John Lamping, Anurag Mendhekar, Chris Maeda, Cristina Lopes, Jean-Marc Loingtier, and John Irwin. Aspect-oriented programming. In *European conference on object-oriented programming*, pages 220–242. Springer, 1997.

[119] Anneke G Kleppe, Jos Warmer, Wim Bast, and MDA Explained. The model driven architecture: practice and promise, 2003.

[120] Handoko Kohar, Philippe Wegner, and Leon Smit. System for setting ambient parameters, September 10 1996. US Patent 5,554,979.

[121] Roland Kossel, Wilhelm Tegethoff, Michael Bodmann, and Nicholas Lemke. Simulation of complex systems using modelica and tool coupling. In *5th Modelica Conference*, volume 2, pages 485–490, 2006.

[122] Michal Kudelski, Luca M Gambardella, and Gianni A Di Caro. Robonetsim: An integrated framework for multi-robot and network simulation. *Robotics and Autonomous Systems*, 61 (5):483–496, 2013.

[123] J. Lawrence. *Introduction to Neural Networks*. California Scientific, Nevada City, CA 95959., 1993. ISBN 1-883157-00-5.

[124] Jonathan Lazar, Alfreda Dudley-Sponaugle, and Kisha-Dawn Greenidge. Improving web accessibility: a study of webmaster perceptions. *Computers in human behavior*, 20(2):269–288, 2004.

[125] Doug Lea and Jos Marlowe. Interface-based protocol specification of open systems using psl. In *European Conference on Object-Oriented Programming*, pages 374–398. Springer, 1995.

[126] Geoffrey Lewis, Steven Barber, and Ellen Siegel. *Programming with Java IDL: Developing Web Applications with Java and CORBA*. John Wiley & Sons, 1998.

[127] Xiaobo Li, Yonglin Lei, Weiping Wang, Wenguang Wang, and Yifan Zhu. A dsm-based multi-paradigm simulation modeling approach for complex systems. In *Simulation Conference (WSC), 2013 Winter*, pages 1179–1190. IEEE, 2013.

[128] Han X Lin, Yee-Yin Choong, and Gavriel Salvendy. A proposed index of usability: a method for comparing the relative usability of different software systems. *Behaviour & information technology*, 16(4-5):267–277, 1997.

[129] Mark W Maier. Architecting principles for systems-of-systems. *Systems Engineering: The Journal of the International Council on Systems Engineering*, 1(4):267–284, 1998.

[130] Anu Maria. Introduction to modeling and simulation. In *Proceedings of the 29th conference on Winter simulation*, pages 7–13. IEEE Computer Society, 1997.

[131] Antoni Martínez-Ballesté, Pablo A Pérez-Martínez, and Agusti Solanas. The pursuit of citizens' privacy: a privacy-aware smart city is possible. *IEEE Communications Magazine*, 51 (6):136–141, 2013.

[132] Nenad Medvidovic and Richard N Taylor. A classification and comparison framework for software architecture description languages. *Software Engineering, IEEE Transactions on*, 26 (1):70–93, 2000.

[133] Mira Mezini, Linda Seiter, and Karl Lieberherr. Component integration with pluggable composite adapters. In *Software Architectures and Component Technology*, pages 325–356. Springer, 2002.

[134] Rim Missaoui, Ghaith Warkozek, Shadi Abras, Stéphane Ploix, and Seddik Bacha. Simulation temps réel pour la gestion des flux énergétiques dans l'habitat. In *IBPSA*, 2010.

[135] M. Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, 1998. ISBN 978-0-262-63185-3.

[136] DEFENSE MODELING. Simulation office. *High Level Architecture for Modeling and Simulation Management Plan*, 1995.

[137] Guillaume Moreau. *Modélisation du comportement pour la simulation interactive: application au trafic routier multimodal*. PhD thesis, 1998.

[138] Yassine Motie, Alexandre Nketsa, and Philippe Truillet. A co-simulation framework interoperability for Neo-campus project (regular paper). In *European Simulation and Modelling Conference (ESM), Lisbon, 25/10/2017-27/10/2017*. EUROSIS, 2017.

[139] Yassine Motie, Elhadi Belghache, Alexandre Nketsa, and Jean-Pierre George. Interoperability based dynamic data mediation using adaptive multi-agent systems for co-simulation. In *2018 International Conference on High Performance Computing & Simulation (HPCS)*, pages 235–241. IEEE, 2018.

[140] Yassine Motie, Alexandre Nketsa, and Philippe Truillet. An assistance tool to design interoperable components for co-simulation. In *International conference on the Sciences of Electronics, Technologies of Information and Telecommunications*, pages 494–503. Springer, 2018.

[141] Michael J Muller and Sarah Kuhn. Participatory design. *Communications of the ACM*, 36 (6):24–28, 1993.

[142] Snehal Mumbaikar, Puja Padiya, et al. Web services based on soap and rest principles. *International Journal of Scientific and Research Publications*, 3(5):1–4, 2013.

[143] Tadao Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.

[144] Xiao Nie. Constructing smart campus based on the cloud computing platform and the internet of things. In *Proceedings of the 2nd International Conference on Computer Science and Electronics Engineering (ICCSEE 2013), Atlantis Press, Paris, France*, pages 1576–1578, 2013.

[145] Jakob Nielsen. *Designing web usability: The practice of simplicity*. New Riders Publishing, 1999.

[146] Jean-François Nogier. *Ergonomie du logiciel et design web-4e éd.: Le manuel des interfaces utilisateur*. Dunod, 2008.

[147] Christian Noll, Torsten Blochwitz, Thomas Neidhold, and Christian Kehrer. Implementation of modelisar functional mock-up interfaces in simulationx. In *Proceedings of the 8th International Modelica Conference; March 20th-22nd; Technical Univeristy; Dresden; Germany*, number 63, pages 339–343. Linköping University Electronic Press, 2011.

[148] David D. Nolte. The tangled tale of phase space. *Physics Today*, 63(4):33–38, April 2010. ISSN 0031-9228, 1945-0699. doi: 10.1063/1.3397041.

[149] Eric Noulard, Jean-Yves Rousselot, and Pierre Siron. Certi, an open source rti, why and how. 2009.

[150] Natalya F Noy, Deborah L McGuinness, et al. Ontology development 101: A guide to creating your first ontology, 2001.

[151] Luke Olsen, Faramarz F Samavati, Mario Costa Sousa, and Joaquim A Jorge. Sketch-based modeling: A survey. *Computers & Graphics*, 33(1):85–103, 2009.

[152] Ekaterina Olshannikova, Aleksandr Ometov, Yevgeni Koucheryavy, and Thomas Olsson. Visualizing big data with augmented and virtual reality: challenges and research agenda. *Journal Of Big Data*, 2(1):1–27, 2015.

[153] Ernest H Page, Richard Briggs, and John A Tufarolo. Toward a family of maturity models for the simulation interconnection problem. In *Proceedings of the Spring Simulation Interoperability Workshop*, volume 1, pages 1059–1069. IEEE Computer Society, 2004.

[154] Peter Palensky, Arjen A Van Der Meer, Claudio David Lopez, Arun Joseph, and Kaikai Pan. Cosimulation of intelligent power systems: Fundamentals, software architecture, numerics, and coupling. *IEEE Industrial Electronics Magazine*, 11(1):34–50, 2017.

[155] Themis Palpanas. Data series management: the road to big sequence analytics. *ACM SIGMOD Record*, 44(2):47–52, 2015.

[156] Hervé Panetto and Arturo Molina. Enterprise integration and interoperability in manufacturing systems: Trends and issues. *Computers in industry*, 59(7):641–646, 2008.

[157] Alexander Paprotny and Michael Thess. *Realtime Data Mining: Self-learning Techniques for Recommendation Engines*. Springer Science & Business Media, 2013.

[158] Gerardo Pardo-Castellote. Omg data-distribution service: Architectural overview. In *Distributed Computing Systems Workshops, 2003. Proceedings. 23rd International Conference on*, pages 200–206. IEEE, 2003.

[159] MG Perhinschi, MR Napolitano, and S Tamayo. Integrated simulation environment for unmanned autonomous systems: towards a conceptual framework. *Modelling and Simulation in Engineering*, 2010:2, 2010.

[160] Jerry Pettersson. Data administration method, August 3 2004. US Patent 6,772,174.

[161] G. Picard and M-P. Gleizes. Cooperative Self-Organization to Design Robust and Adaptive Collectives. In *2nd International Conference on Informatics in Control, Automation and Robotics (ICINCO'05), Barcelona, Spain, Volume I*. INSTICC Press, 2005.

[162] Gauthier Picard, Marie Pierre Gleizes, et al. Cooperative self-organization to design robust and adaptive collectives. In *ICINCO*, pages 236–241. Citeseer, 2005.

[163] Paul Piché, Elhadi Belghache, Nicolas Verstaevel, and Berangere Lartigue. Impact of ecofeedback on the behavior of campus users. *Energy*, page 8, 2014.

[164] J Mark Pullen, Ryan Brunton, Don Brutzman, David Drake, Michael Hieb, Katherine L Morse, and Andreas Tolk. Using web services to integrate heterogeneous simulations in a grid environment. *Future Generation Computer Systems*, 21(1):97–106, 2005.

[165] Davide Quaglia, Riccardo Muradore, Roberto Bragantini, and Paolo Fiorini. A systemc/-matlab co-simulation tool for networked control systems. *Simulation Modelling Practice and Theory*, 23:71–86, 2012.

[166] Gauthier Quesnel, Raphaël Duboz, and Éric Ramat. The virtual laboratory environment–an operational framework for multi-modelling, simulation and analysis of complex dynamical systems. *Simulation Modelling Practice and Theory*, 17(4):641–653, 2009.

[167] Wullianallur Raghupathi and Viju Raghupathi. Big data analytics in healthcare: promise and potential. *Health Information Science and Systems*, 2(1):3, 2014.

[168] Scott A Renner, Arnon S Rosenthal, and James G Scarano. Data interoperability: Standardization or mediation. In *1st IEEE Metadata Conference*. Citeseer, 1996.

[169] Marc Rettig. Prototyping for tiny fingers. *Communications of the ACM*, 37(4):21–27, 1994.

[170] Reza Rezaei, Thiam-kian Chiew, and Sai-peck Lee. A review of interoperability assessment models. *Journal of Zhejiang University SCIENCE C*, 14(9):663–681, 2013.

[171] Peter Riederer, Werner Keilholz, Vanessa Ducreux, and France Antipolis Cedex. Coupling of trnsys with simulink-a method to automatically export and use trnsys models within simulink and vice versa. *Building Simulation, Glasgow, Royaume-Uni*, 2009.

[172] Michael Rohs and R Bohn. Entry points into a smart campus environment-overview of the ethoc system. In *Distributed Computing Systems Workshops, 2003. Proceedings. 23rd International Conference on*, pages 260–266. IEEE, 2003.

[173] Nirmalya Roy, Gautham Pallapa, and Sajal K Das. A middleware framework for ambiguous context mediation in smart healthcare application. In *Wireless and Mobile Computing, Networking and Communications, 2007. WiMOB 2007. Third IEEE International Conference on*, pages 72–72. IEEE, 2007.

[174] Behrokh Samadi. Distributed simulation, algorithms and perfomance analysis. *PhD Thesis, COmputer Science Department, Universiyt of California, Los Angeles*, 1985.

[175] Simon Schenk and Steffen Staab. Networked graphs: a declarative mechanism for sparql rules, sparql views and rdf data integration on the web. In *Proceedings of the 17th international conference on World Wide Web*, pages 585–594. ACM, 2008.

[176] Mathieu Schumann, Benoit Charrier, Gilles Plessis, and Bénédicte Wall-Ribot. Buildsyspro un bibliotheque modelica open source pour l'énergétique des bâtiments et des quartiers. In *Conférence IBPSA France, Marne la Vallée, France*, 2016.

[177] Reinhard Sefelin, Manfred Tscheligi, and Verena Giller. Paper prototyping-what is it good for?: a comparison of paper-and computer-based low-fidelity prototyping. In *CHI'03 extended abstracts on Human factors in computing systems*, pages 778–779. ACM, 2003.

[178] Ziad Sehili, Lars Kolb, Christian Borgs, Rainer Schnell, and Erhard Rahm. Privacy preserving record linkage with ppjoin. In *BTW*, pages 85–104, 2015.

[179] Marcos Serrano and Laurence Nigay. Openwizard: une approche pour la création et l'évaluation rapide de prototypes multimodaux. In *Proceedings of the 21st International Conference on Association Francophone d'Interaction Homme-Machine*, pages 101–109. ACM, 2009.

[180] Giovanna Di Marzo Serugendo, Marie-Pierre Gleizes, and Anthony Karageorgos. *Self-organising software: From natural to artificial adaptation.* Springer Science & Business Media, 2011.

[181] Julien Siebert. *Approche multi-agent pour la multi-modélisation et le couplage de simulations. Application à l'étude des influences entre le fonctionnement des réseaux ambiants et le comportement de leurs utilisateurs.* PhD thesis, Université Henri Poincaré-Nancy I, 2011.

[182] Jon Siegel and Dan Frantz. *CORBA 3 fundamentals and programming*, volume 2. John Wiley & Sons New York, NY, USA:, 2000.

[183] MATLAB Simulink and Model Based Design. The mathworks inc. *Ver*, 7(4.365):R14, 1995.

[184] J. Smith and J. Doe. Current advances in technology. In *22nd CCOPT*, pages 427–437. ACM, 1990.

[185] Roger D Smith. Closing the gap between simulation & combat computer systems. In *Proceedings of the 28th conference on Winter simulation*, pages 939–945. IEEE Computer Society, 1996.

[186] Carolyn Snyder. *Paper prototyping: The fast and easy way to design and refine user interfaces.* Morgan Kaufmann, 2003.

[187] Srinath Srinivasa and Sameep Mehta. *Big Data Analytics: Third International Conference, BDA 2014, New Delhi, India, December 20-23, 2014. Proceedings*, volume 8883. Springer, 2015.

[188] Paul W Sulton. Interoperability: A new paradigm. *Computational Intelligence for Modelling, Control & Automation: Neural Networks & Advanced Control Strategies*, 3:351, 1999.

[189] Junichi Suzuki and Yoshikazu Yamamoto. Toward the interoperable software design models: quartet of uml, xml, dom and corba. In *Software Engineering Standards, 1999. Proceedings. Fourth IEEE International Symposium and Forum on*, pages 163–172. IEEE, 1999.

[190] S. Svensson. *Introduction to technology.* Publishing Press, 1996. ISBN 0-0000-0000-0.

[191] Clemens Szyperski. Independently extensible systems-software engineering potential and challenges. *Australian Computer Science Communications*, 18:203–212, 1996.

[192] Lu Tan and Neng Wang. Future internet: The internet of things. In *Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference on*, volume 5, pages V5–376. IEEE, 2010.

[193] Simon JE Taylor, Xiaoguang Wang, Stephen John Turner, and Malcolm Yoke-Hean Low. Integrating heterogeneous distributed cots discrete-event simulation packages: an emerging standards-based approach. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 36(1):109–122, 2006.

[194] Michael M Tiller. Physical modeling with modelica, 2001.

[195] Andreas Tolk. Moving towards a lingua franca for m&s and c3i–developments concerning the c2iedm. In *European Simulation Interoperability Workshop*, pages 268–275, 2004.

[196] Andreas Tolk. Xml mediation services utilizing model based data management. In *Simulation Conference, 2004. Proceedings of the 2004 Winter*, volume 2, pages 1476–1484. IEEE, 2004.

[197] Andreas Tolk and James A Muguira. The levels of conceptual interoperability model. In *Proceedings of the 2003 fall simulation interoperability workshop*, volume 7, pages 1–11. Citeseer, 2003.

[198] Andreas Tolk and J Mark Pullen. Using web services and data mediation/storage services to enable command and control to simulation interoperability. In *Distributed Simulation and Real-Time Applications, 2005. DS-RT 2005 Proceedings. Ninth IEEE International Symposium on*, pages 27–34. IEEE, 2005.

[199] Stavros Tripakis. Bridging the semantic gap between heterogeneous modeling formalisms and fmi. In *Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS)*, pages 60–69. IEEE, 2015.

[200] Hong-Linh Truong and Thomas Fahringer. Performance analysis, data sharing, and tools integration in grids: New approach based on ontology. *Computational Science-ICCS 2004*, pages 424–431, 2004.

[201] Chun-Wei Tsai, Chin-Feng Lai, Han-Chieh Chao, and Athanasios V Vasilakos. Big data analytics: a survey. *Journal of Big Data*, 2(1), 2015.

[202] Antonio Vallecillo, Juan Hernández, and José M Troya. New issues in object interoperability. In *European Conference on Object-Oriented Programming*, pages 256–269. Springer, 2000.

[203] Herman Van der Auweraer, Jan Anthonis, Stijn De Bruyne, and Jan Leuridan. Virtual engineering at work: the challenges for designing mechatronic products. *Engineering with Computers*, 29(3):389–408, 2013.

[204] Hans Vangheluwe et al. Devs as a common denominator for multi-formalism hybrid systems modelling. In *IEEE international symposium on computer-aided control system design*, volume 134. IEEE, 2000.

[205] Julien Vaubourg, Yannick Presse, Benjamin Camus, Laurent Ciarletta, Vincent Chevrier, Jean-Philippe Tavella, Boris Deneuville, and Olivier Chilard. Simulation de smart grids avec mecsyco. *23es Journées Francophones sur les Systèmes Multi-Agents (JFSMA'15)*, pages 217–218, 2015.

[206] Nicolas Verstaevel, Christine Régis, Marie-Pierre Gleizes, and Fabrice Robert. Principles and experimentations of self-organizing embedded agents allowing learning from demonstration in ambient robotic (regular paper). In *International Conference on Ambient Systems, Networks and Technologies (ANT), London, United Kingdom*. Elsevier, 2015.

[207] Nicolas Verstaevel, Jérémy Boes, and Marie-Pierre Gleizes. From smart campus to smart cities: Issues of the smart revolution. 2017.

[208] Nicolas Verstaevel, Jérémy Boes, Julien Nigon, Dorian D'Amico, and Marie-Pierre Gleizes. Lifelong Machine Learning with Adaptive Multi-Agent Systems (regular paper). In *International Conference on Agents and Artificial Intelligence (ICAART)*, volume 2, pages 275–286. SciTePress, 2017.

[209] Gottfried Vossen. Big data as the new enabler in business and other intelligence. *Vietnam Journal of Computer Science*, 1(1):3–14, 2014.

[210] Wenguang Wang, Andreas Tolk, and Weiping Wang. The levels of conceptual interoperability model: applying systems engineering principles to m&s. In *Proceedings of the 2009 Spring Simulation Multiconference*, page 168. Society for Computer Simulation International, 2009.

[211] G Wei et al. Multiagent systems, a modern approach to distributed artificial systems. 1999.

[212] G. Weiß. *Multiagent Systems, A modern Approach to Distributed Artificial Systems*. MIT Press, 1999.

[213] Michael Wetter. Co-simulation of building energy and control systems with the building controls virtual test bed. *Journal of Building Performance Simulation*, 4(3):185–203, 2011.

[214] Michael Wetter, Thierry S Nouidui, David Lorenzetti, Edward A Lee, and Amir Roth. Prototyping the next generation energyplus simulation engine. In *Proceedings of the 3rd IBPSA Conference, Jeju island, South Korea*, pages 27–29, 2015.

[215] Stephen A White. *BPMN modeling and reference guide: understanding and using BPMN*. Future Strategies Inc., 2008.

[216] Edmund Widl, Wolfgang Müller, Atiyah Elsheikh, Matthias Hörtenhuber, and Peter Palensky. The fmi++ library: A high-level utility package for fmi for model exchange. In *Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES), 2013 Workshop on*, pages 1–6. IEEE, 2013.

[217] Gio Wiederhold. Mediators in the architecture of future information systems. *Computer*, 25 (3):38–49, 1992.

[218] Gio Wiederhold. Mediation in information systems. *ACM Computing Surveys (CSUR)*, 27 (2):265–267, 1995.

[219] Sara Williams and Charlie Kindel. The component object model: A technical overview. *Dr. Dobbs Journal*, 356:356–375, 1994.

[220] Annette L Wilson and Richard M Weatherly. The aggregate level simulation protocol: an evolving system. In *Proceedings of the 26th conference on Winter simulation*, pages 781–787. Society for Computer Simulation International, 1994.

[221] Daniel M Yellin and Robert E Strom. Protocol specifications and component adaptors. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 19(2):292–333, 1997.

[222] Sungjoo Yoo and Kiyoung Choi. Optimistic timed hw-sw cosimulation. In *in Proc. of APCHDL'97*. Citeseer, 1997.

[223] D. Young. Beyond today's technology. In *CPPTP 2003*, pages 427–437. IEEE, 2003.

[224] Zhiwen Yu, Yunji Liang, Bukan Xu, Yue Yang, and Bin Guo. Towards a smart campus with mobile social networking. In *Internet of Things (iThings/CPSCom), 2011 international conference on and 4th international conference on cyber, physical and social computing*, pages 162–169. IEEE, 2011.

[225] Gregory Zacharewicz. *Un environnement G-DEVS/HLA: Application à la modélisation et simulation distribuée de workflow*. PhD thesis, 2006.

[226] Gregory Zacharewicz, Saikou Diallo, Yves Ducq, Carlos Agostinho, Ricardo Jardim-Goncalves, Hassan Bazoun, Zhongjie Wang, and Guy Doumeingts. Model-based approaches for interoperability of next generation enterprise information systems: state of the art and future challenges. *Information Systems and e-Business Management*, 15(2):229–256, 2017.

[227] Amy Moormann Zaremski and Jeannette M Wing. *Specification matching of software components*, volume 20. ACM, 1995.

[228] Bernard P Zeigler. Devs representation of dynamical systems: Event-based intelligent control. *Proceedings of the IEEE*, 77(1):72–80, 1989.

[229] Bernard P Zeigler. *Object-oriented simulation with hierarchical, modular models: intelligent agents and endomorphic systems*. Academic press, 2014.

[230] Bernard P Zeigler, Herbert Praehofer, and Tag Gon Kim. *Theory of modeling and simulation: integrating discrete event and continuous complex dynamic systems*. Academic press, 2000.

[231] Kun Zhang, Wei Fan, Xiaojing Yuan, Ian Davidson, and Xiangshang Li. Forecasting skewed biased stochastic ozone days: Analyses and solutions. In *Proceedings of the Sixth International Conference on Data Mining*, ICDM '06, pages 753–764, Washington, DC, USA, 2006. IEEE Computer Society. ISBN 0-7695-2701-9. doi: 10.1109/ICDM.2006.73.

[232] Zhi Zhang, Zhonghai Lu, Qiang Chen, Xiaolang Yan, and Li-Rong Zheng. Cosmo: Co-simulation with matlab and omnet++ for indoor wireless networks. In *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, pages 1–6. IEEE, 2010.

[233] J Zueco and A Campo. Network model for the numerical simulation of transient radiative transfer process between the thick walls of enclosures. *Applied Thermal Engineering*, 26(7): 673–679, 2006.

[234] John Zukowski. The java compiler api. *Java^{TM} 6 Platform Revealed*, pages 155–169, 2006.