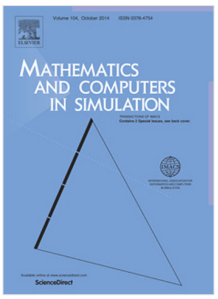
Accepted Manuscript



Simulation based design: Overview about related works

Wafa Mefteh

PII: S0378-4754(18)30073-9
DOI: https://doi.org/10.1016/j.matcom.2018.03.012
Reference: MATCOM 4559

To appear in: *Mathematics and Computers in Simulation*

Received date : 19 March 2017
Revised date : 13 November 2017
Accepted date : 17 March 2018

Please cite this article as: W. Mefteh, Simulation based design: Overview about related works, *Math. Comput. Simulation* (2018), https://doi.org/10.1016/j.matcom.2018.03.012

Simulation Based Design: Overview about Related Works

Wafa MEFTEH

MIRACL Laboratory, University of Sfax, Tunisia

Abstract

Simulation is a very important tool which has been used for several years in several fields (with computer and non computer systems). Given its importance, simulation has been used as soon as possible during a system's design phase. This paradigm is called "Simulation-Based Design" or SBD. Using SBD aims at eliminating unfit designs as early as possible before significant resources have been consumed. It is the process in which simulation is the primary means of evaluation and verification. In this paper, we give an overview about related works. We classify the related work in three classes: simulation to design non computer systems in several domains, simulation to design software and simulation to design Multi-Agent Systems, self-Organising Systems or Systems with Emergent Functionality.

Keywords: Simulation, Simuation-Based Design, Complex System Design, Computer Based Simulation, Mathematics Based Simulation

1. Introduction

To understand and analyze a phenomenon or to anticipate the expected results, several means and tools have been used for long. Among these means and tools, we can cite the following:

- Replacing humans by animals: this consists in finding the animal populations whose behaviors are similar to human behaviors with respect to a given phenomenon. Many pressure groups are fighting against this practice.

- Man-oeuvres: for example for the military whose play two opposing forces (the white against the black) on a real course with real hardware but without using live ammunition.
- Mock-up: this consists in building a copy with small scale and testing it.
- Equation: there are many examples which have been used by pupils and students in courses of physics, chemistry, etc... This simulation can be used only with simple phenomena.

All these tools are simulation. The main purpose of simulation is to shed light on the underlying mechanisms that control the behavior of a system. It is practically used to understand and predict the behavior of the system. So to predict the way in which the system will evolve and respond to its surroundings, so we can identify any necessary changes that will help to make the system performs the way that we want it to [21]. Given its importance, simulation has been used as soon as possible from the design process in order to minimize design costs and eliminate unfit designs as early as possible. This paradigm called "Simulation-Based Design" or SBD has been applied in several areas especially with systems that require a thorough study of what the system can deliver before consuming enormous resources to eliminate as soon as possible design failures in order to save effort, time and resources.

In this paper, we give an overview about related works where SBD is used. Our objective is not to give a state of the art about all existing works. The purpose is mainly to see how simulation has been used in several domains. What techniques are used for different types of systems ? What tools have been used ?

For this, we decomposed the related works into three parts. We firstly talk about using simulation to design non computer systems in several domains, then using simulation to design software and finally to design Multi-Agent Systems (MAS), self-organizing systems and systems with emergent functionality.

2. Simulation Based Design for Non-Computer Systems

Simulation Based Design has been applied in several areas such as electronic, automotive, maritime, product life cycle, fine chemistry and phar-

macy, production, plant layout, shipboard power system protection... . In this section, we present an overview about these works.

Simulation has been largely used to design supply chain which is a complex process integrating several actors (suppliers, manufacturers, warehouses, and retailers). The main purpose of a supply chain process is to produce goods and deliver them at the right quantities, and at the right time, while minimizing costs as well as satisfying customer requirements. The difficulty with supply chain modeling is that a poor plan can easily propagate to the whole supply chain areas [10]. Some solutions were proposed as *the Enterprise Resource Planning* and *the Supply Chain Management* but it is too costly to use those solutions for academic research.
[5], [9] and [6] discuss the use of simulation to model supply chain. They aim at taking advantages from its benefits for a specific supply chain problems. They demonstrate that using simulation permits to enhance the quality of the produced supply chain with minimal costs. Indeed, simulation:

- helps to understand the overall supply chain processes and characteristics by graphics/animation;
- permits to capture system dynamics: using probability distribution, user can model unexpected events in certain areas and understand the impact of these events on the supply chain;
- minimize the risk of changes in planning process: by *what-if* simulation which enables the user testing various alternatives before changing plan.

These researchers do not propose a simulation-based approaches to model supply chain. They only give demonstrations of the benefits from using simulation to design supply chain. Other researchers propose simulation approaches for supply chain modeling and develop simulation models and simulation-modeling tools to address different needs within supply chain domains. Biswas et al. [30], developed DESSCOM, an object oriented supply chain simulation modeling methodology. Figure 1 presents the architecture of DESSCOM which details the steps followed using DESSCOM. These steps are:

- Specify supply chain: the construction of the object models.

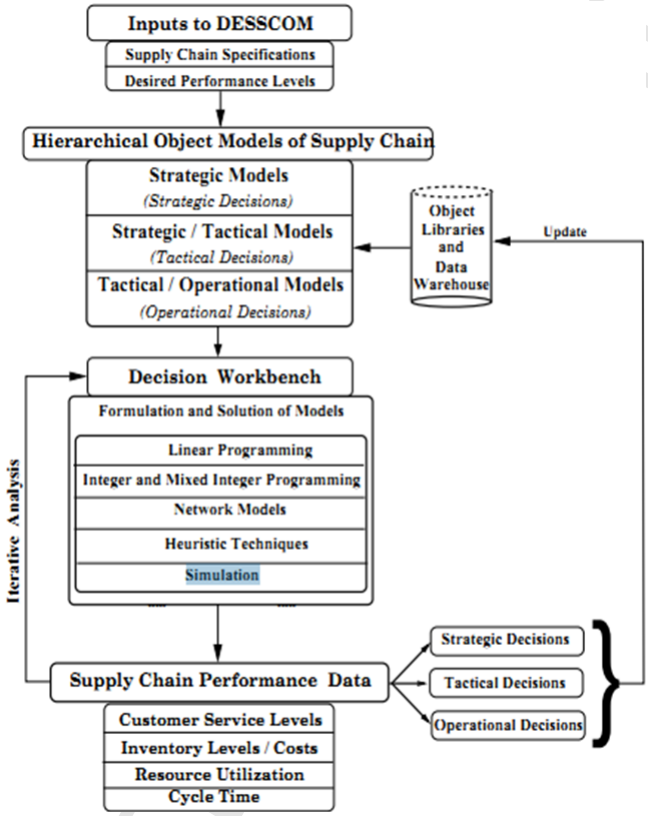


Figure 1: The DESSCOM architecture proposed by Biswas et al. [30] as an object oriented supply chain simulation modeling methodology.

- Generate problem formulation: the object model created in the previous step is then used to generate the data for an analysis or an optimization model of the supply chain.
- Solve problem: the problem formulation generated above is then solved using an appropriate methodology.
- Iterate if necessary: this is realized using an analytical model to obtain optimal parameters at an aggregate level and then using simulation to refine these parameters to take into account the details.

This methodology contains iterative activities based on simulation and analysis. The iterative analysis enables a progressive construction of the model which is important for the final decision making.

Cope et al. [12] propose an approach that provides a simulation solution for supply chain modeling. An overview of this methodology is presented in figure 2. It consists in the definition (by users) of the supply chain simulation model using SCOR [1] (Supply Chain Operations Reference) based on ontology. The ontology will include supply chain knowledge (supply chain elements, functional units, processes, information, etc) and the knowledge required to build a simulation model of the supply chain system. The simulation model will then be generated automatically from the ontology. The

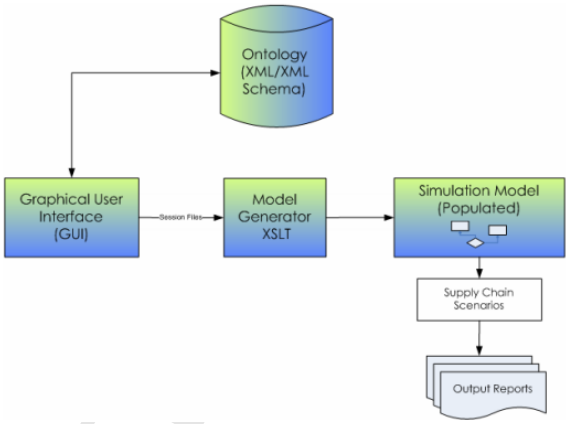


Figure 2: A hybrid modeling framework for supply chain simulation proposed by Cope et al. [12] as a simulation solution for supply chain modeling.

main goal of this automatic simulation model generator is to provide sound tools for the end users to input their logistics structures and interactions accurately without requiring too much knowledge of simulation techniques. The automatic generation is a good point which reduces the simulation modeling cycle time. Simulation has been a very effective tool in cast components design. The foundry engineer participates early in the product development stage. This reduces the time between the concept stage and the production stage.

In this area, [2] and [17] demonstrate the importance of using simulation

[1] It is a common model for defining supply chains. SCOR model is developed by the supply chain council and aims to describe the operations of various supply chain constructs. It classifies the operations of supply chain as Plan, Source, Make, Deliver and Return.

to design cast components. They present the advantages of using simulation approach regarding the old trial and error approach for process development. They highlight some case studies for casting process. The advantages are:

- Simulation helps to calculate real costs.
- Simulation can be used as a tool to negotiate the right quality.
- The modeling has been made easy for the designer.
- Decreasing the modeling errors and costs. Authors indicate that the costs to change the design increase ten-fold in every step of the design and manufacturing process.
- Simulation has also proven to be an effective educational tool in the foundry industry.

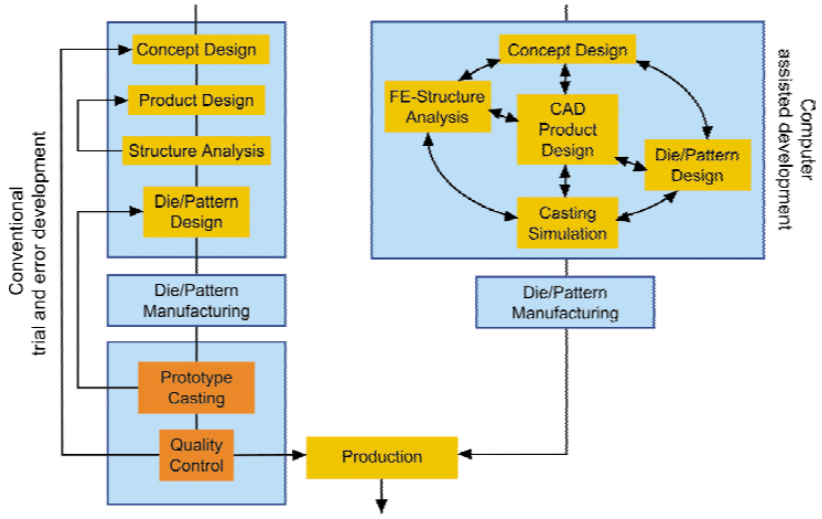


Figure 3: Trial and error development versus Computer Assisted Development as presented in [17] to highlight the advantages of using simulation approach regarding the old trial and error approach for process development.

Figure 3 presents the trial and error development versus Computer Assisted Development. This figure shows that with the first development, the control is made after the realization of the activity which can necessitate to return to the previous activities in order to make changes and corrections. With

the new development, using simulation, the different activities are verified progressively during their realization which enables the production of the right quality with less cost than the first way because it enables less mistakes. The goal of this research is essentially to demonstrate the benefits from using simulation to design cast components.

Simulation has been also used to design plant layout. There are many techniques to design plant layout but their results are limited and layout design and planning is becoming more and more critical due to shorter product life cycles and highly dynamic demand conditions. M Arni Luovo [1] has demonstrated that, the traditional layout types such as process layout and product layout do not have the ability to respond quickly to the changes.

Jayachitra et al. *[22]* use the WITNESS 2006 simulation software to simulate many models. Design of experiments is used to plan the simulation experiment. They analyze the different results in order to identify the suitability of a particular layout in a given environment.

Smutkupt et al. *[32]* use simulation in the plant layout design to show more information about the design such as total time in system, waiting and use time (figure 5). They indicate that the result from the other popular techniques to design industrial plant layout, like CRAFT (Computerize Relative Allocation Facilities Technique), was limited and showed only minimum total transfer cost between departments. Thus, they use Microsoft Visual Basic to develop a design system based on CRAFT model. Then, it is used to link the design system to a simulation system in Arena as represented in figure 4. Arena is a simulation software from Rockwell Software Company. Finally, the simulation system sends back overall results to a report system in Microsoft Visual Basic output form.

This approach considers a preliminary validation before experimenting the design. The experimentation results are analysed in order to decide if a new experimentation is needed or not.

Michael Jensen *[23]* applies simulation to design hybrid vehicle in order to optimize the performance, reliability and cost. The hybrid vehicle were advanced as the bridge between the internal combustion engine and increased tailpipe emissions. They offer the increased fuel efficiency and reduced emissions of the electric vehicle, and the long distance range and readily available

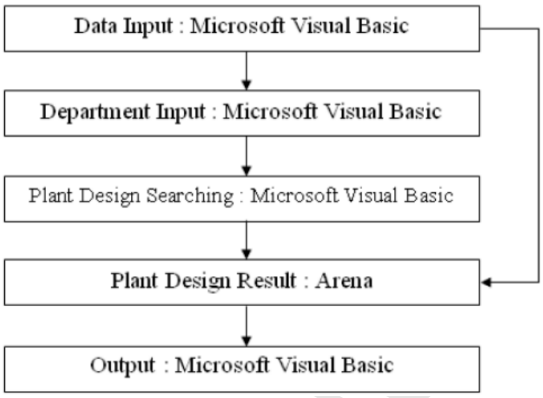


Figure 4: Plant layout design simulation with ARENA presented by Smutkupt et al. [32] to link the design system to a simulation system in Arena.

support infrastructure of an internal combustion engine vehicle. In a hybrid vehicle, the drive contains components of both internal combustion engine and electric vehicles. Their integral use of mechanical, electrical and software technologies makes system integration more challenging and the design process more complex than for conventional vehicles. For this, Michael Jensen proposes a SBD methodology (figure 6) to design hybrid vehicle systems. The objective from proposing this methodology is making the design process of hybrid vehicle less complex. The methodology considers different levels of verification. It contains also activities of analysis and evaluation. The problem is that we don't see any iteration in figure 6. Generally, verification, analysis and evaluation activities are followed by returns to previous activities in order to make necessary changes if the result of these activities is insufficient.

Maarten Chin Seah et al. [11] propose an agent-based modeling and simulation approach for design and analysis of NASA's Mars Exploration Rover (MER) mission operations. A space mission operations system is a complex network of human organizations, information and deep space network systems and spacecraft hardware. This approach is based on Brahms which is a multi-agent modeling and simulation tool. Brahms allows simulation of these elements of the space mission operation as agents and objects with behaviors. This approach is based on multi-agent simulation which enable the decomposition of the system into less complex entities. The main

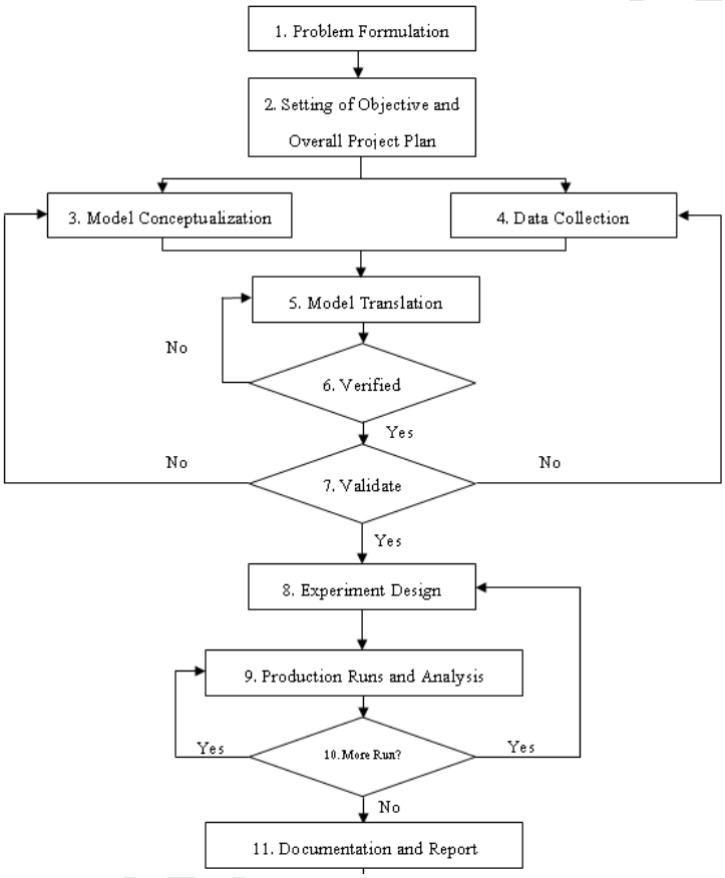


Figure 5: The simulation technique of developed by Smutkupt et al[32] to introduce simulation in the plant layout design.

advantage of this approach is that when an error occurs, it is more simple to repair it than a non decomposed simulation and thus this decreases the design cost.

As a synthesis, from the study of several works of which the main ones were cited above and which studying and using simulation to design non-computer systems, we deduce that simulation has been greatly used to design these systems because they require a thorough study of what the system can deliver before consuming enormous resources in order to eliminate as soon as possible design failures and so save effort, time and resources. The approaches are classified in two principal classes as mentioned in table 1: model-based

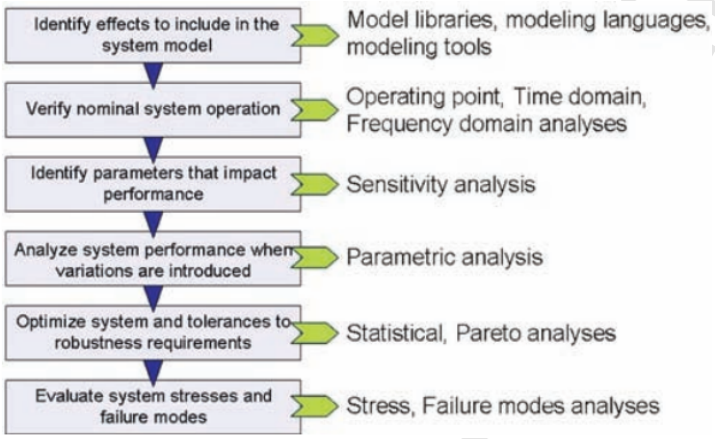


Figure 6: Hyprid vehicle system design process depending on a systematic development flow and requires advanced simulation capabilities developed by Michael Jensen [23] to use simulation to design hybrid vehicle.

simulation class and test-based simulation class.

3. Simulation-Based Design for Software

Simulation Based Design for software has been essentially used to cope with the complexity of the development of some software.

Xiaolin Hu [34] presents a simulation-based software development methodology to manage the complexity of distributed real time software (figure 7). This methodology, based on discrete event system specification (DEVS), has been proposed to overcome the "incoherence problem" between different design stages by emphasizing "model continuity" through the development process. Specifically, the designed control models can be tested and analyzed by simulation methods and then deployed to the distributed target system for execution. They developed a virtual test environment which enables software to be effectively tested and analyzed in a virtual environment, using virtual sensor/actuators. Within this environment, step-wise simulation methods have been developed so these different aspects, such as logic and temporal behaviors, of a real time system can be tested and analyzed incrementally. This methodology proposed three simulation methods to be used to simulate different aspects of real time systems. Based on the simulation results, the model can be refined if necessary.

Simulation-Based Design for Non Computer Systems	
Model-based simulation	**Test-based simulation**
The model-based simulation consists in the simulation of an image of the system which is usually incomplete because the designer does not have complete knowledge about the domain of the system. In this case, simulation gives the designer more information about the system and the domain and thus enables the possibility of completing the missing characteristics.	The test-based simulation consists in the simulation of an image of the system. This image is usually almost complete because the designer can have complete knowledge about the system domain but he wants to simulate the functioning of the system before building the real system. We are not talking here about the difference between test and simulation but we are talking about two types of simulation. The test is another paradigm which differs from simulation. It consists in the experimentation of the finished real system in order to ensure its proper function.

Table 1: Main classes of works applying SBD for Non Computer Systems.

Ehsan Azarnasab et al. *[3]* present a progressive SBD methodology for developing complex systems. It provides a design process that explicitly focuses on systematic transitions from simulation models to real system realization. The design process consists of three stages which are depicted in figure 8, each of which is characterized by the types of entities (virtual or real) that are involved.

- The first stage is conventional simulation, where simulation is carried out using all models.
- The second stage is virtual environment simulation, where simulation-based study is carried out with combined real system components and simulation models.
- The final stage is real system test, where all real system components

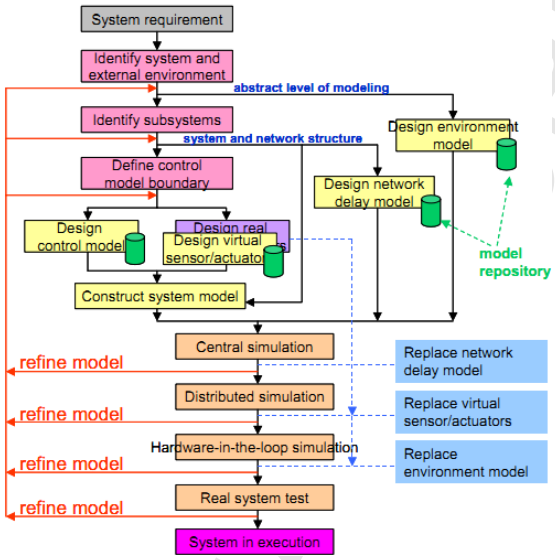


Figure 7: The simulation-based software development methodology proposed by Xiaolin Hu [34] to manage the complexity of distributed real time software.

are tested in a physical environment.

Along this design process, the framework emphasizes two parallel activities in a progressive manner: replace models with real system components, and update models. As the design moves forward, real system components are gradually brought into the simulation to replace models. This progressive SBD methodology was applied to design a networked software (radio system). This software refers to a radio communication system capable of transmitting and receiving different modulated signals across a large frequency spectrum using software programmable hardware. This methodology enables designers to validate their design assumptions and to reveal new design details overlooked before. Such information is fed backward to the previous stages to update the models if needed. The updated model will then be used for follow-on design and test. This activity of model update enables designers to maintain a coherent model of the system under development.

Simonetta Balsamo et al. *[4]* give a simulation-based performance modeling of software architectures specified in UML. Software performance is the process of predicting and evaluating whether the software system satisfies performance goals defined by the user. Performance evaluation refers to the

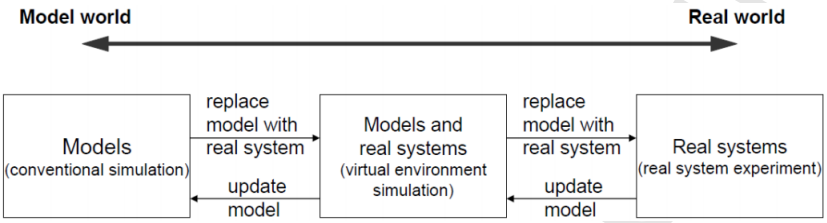


Figure 8: The development process of the methodology developed by Ehsan Azarnasab et al. [3] for developing complex systems.

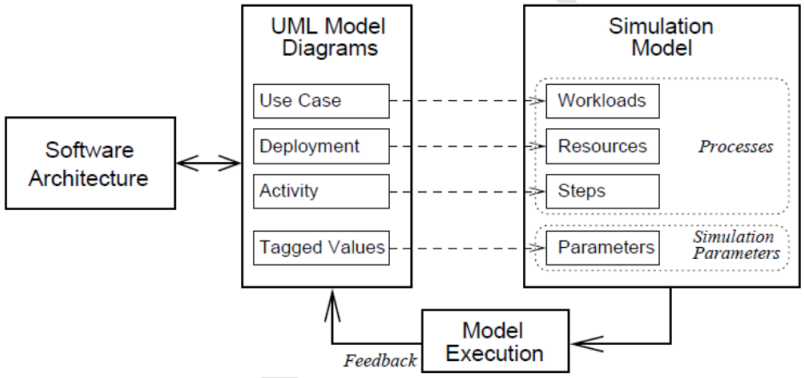


Figure 9: The methodology proposed by Simonetta Balsamo et al. [4] for simulation modeling of software systems and their performance.

activity of measuring the performances of an actual implementation of a system. The software architecture specification is considered as a set of annotated UML Use Cases, Activity and Deployment diagrams. Annotations are expressed according to a subset of the UML Performance Profile. Simulation model parameters are derived from the tagged values extracted from the UML diagrams. Once the process structure of the simulation model is defined, its implementation is built into a simulation program, which is eventually executed. Simulation results are inserted into the UML diagrams as tagged values and can be used to provide a feedback at the software design level. The modeling cycle can be iterated to compare design alternatives and to identify a software architecture that satisfies given performance requirements. This approach (figure 9) translates Use Case diagrams into processes called workloads in the simulation model, Activity diagrams into processes called generic processing steps, and Deployment diagrams into processes representing computational resources (processors). Simulation model

parameters are derived from the tagged values extracted from the UML diagrams representing the software architecture. Once the process structure of the simulation model is defined, its implementation is built into a simulation program, which is eventually executed. The main objective of this approach is the simulation of the system's performance. The problem with this methodology is that the translation of the UML diagrams to the simulation model is made manually.

Gabriel Wainer et al. [33] propose DEMES (Discrete-Event Modeling of Embedded Systems) which is a modeling and simulation-based development methodology based on discrete-event systems specifications. Figure 10 shows the architecture of the process used in DEMES. SoI is the system of interest to be modeled and simulated. This process consists of:

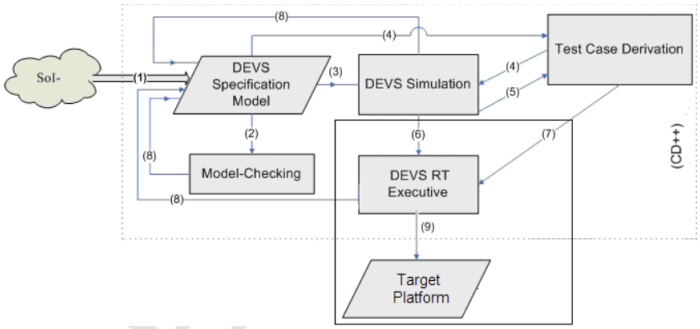


Figure 10: DEMES development cycle proposed by Gabriel Wainer et al. [33] for modeling and simulation-based development of software.

- (1): Defining a specification model of the system using a formal model (DEVS or alternative techniques translated to equivalent DEVS models).
- (2): Model-checking can be used for validation of the model properties.
- (3): The same models are then used to run DEVS simulations of the behavior of the different sub-models under specific loads
- (4) and (5): The same DEVS specification model is used to derive test cases, which can be also used for the simulation studies. Test cases are derived from both the model and from the simulation results in order to check that the model conforms to the requirements.

- (6) If the designer is satisfied with both analytic and simulated results, the model is incrementally moved into a target platform. A real-time Executive (6) executes the models on the particular hardware (9). The executive allows to execute dynamic models and to schedule static and dynamic tasks.

- (7): The parts that are still unverified in the formal and simulated environments are tested. Some parts can still unverified when the hardware is not available and so the software components still be developed incrementally and tested against a model of the hardware to take early design decisions. As the design process evolves, both software and hardware models can be refined..

- (8): Any modifications require going back to the same model specifications.

The principal characteristic of this approach (DEMES) is that it combines the advantages of M&S (Modeling and Simulation) with the rigor of a formal methodology based on DEVS (Discrete Event Systems Specification) formalism. The software life-cycle is cyclic, allowing refinement following a spiral approach.

As a synthesis, software design saw more evolution in recent years because of the diversity of application areas. Systems are becoming increasingly complex, distributed and dynamic. To meet the current systems complexity, efficient technologies should be used to better develop and explore these systems before implementation. SBD was greatly applied to support software products. From the study of related works, we classify existing approaches in two main classes as mentioned in table 2.

Note that, given the popularity of UML, researchers always think to reuse it or inspire from its diagrams and protocols even if it does not allow total design of their systems and they will be forced to make extensions to meet the specificity of their systems. Reusing standards and extending them is good in the sense that it favors the standardization of the new proposed methods based on these standards.
For the second class, we find that during the last years, discrete event simulation has been largely adopted. Discrete event modeling and simulation consists in modeling and simulating systems based on the DEVS (Discrete

Simulation Based Design for Software	
Extending UML	**New approaches without referring to UML**
Reuse UML paradigm. They make extensions to UML diagrams and integrate simulation-based activities.	They present a new way (a specific process) integrating simulation techniques in order to apply SBD without reference to UML.

Table 2: Main classes of works applying SBD to develop Software.

EVent Specification) formalism. A real system modeled with DEVS is described as a hierarchical and modular composite of models that can be behavioral (atomic) or structural (coupled) [33] [14].

4. Simulation-Based Design for Multi Agent Systems, Self-Organizing Systems and Systems with Emergent Functionality

We begin this section by giving a brief definition of Multi Agent Systems, self-organizing systems and systems with emergent functionality. A Multi-Agent System is (MAS) is a set of software agents that interact to solve problems that are beyond the individual (agent) capacities or knowledge of each problem solver. A self-organizing system is a structure that processes where some form of coordination which arises out of the local interactions between its component parts initially disordered. The process of self-organization can be spontaneous. System with emergent functionality tends to be more tolerant because they are more adaptable when their environment change.
In this section we are interested to works which use simulation to facilitate the design of such systems. Why simulation has been used to design such systems and what are the adopted approaches.

Fortino et al. [19], [13] propose an agent-oriented simulation-driven development process obtained by enriching the PASSI [2] methodology with

[2]Process for Agent Societies Specification and Implementation

a simulation step supported by MASSIMO [3], a Java-based discrete event simulation framework for MAS based on a simulation methodology. PASSI methodology is a step-by-step requirements-to-code methodology for designing and developing multi-agent societies. Figure 11 presents the simulation methodology integrated to PASSI. This simulation methodology is based on the following three iterate phases: modeling, coding and simulation of the MAS under-development.

- The Modeling phase is enabled by the Distilled State Charts (DSCs) formalism.
- The Coding (or prototyping) phase is supported by the Mobile Active Object Framework (MAO Framework).
- The simulation phase is supported by MASSIMO framework.

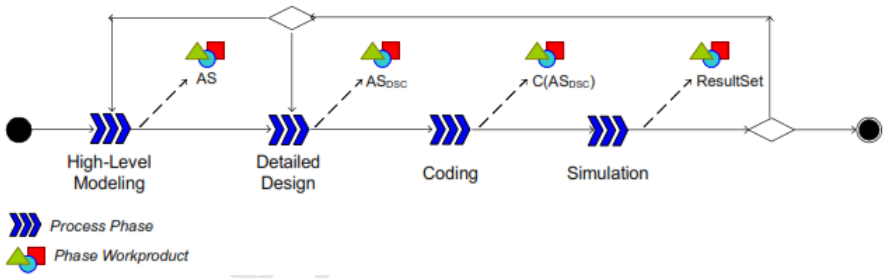


Figure 11: The simulation methodology proposed by Fortino et al. [19], [13] and integrated into the PASSI methodology

The Simulation phase of MASSIMO is to allow the validation and evaluation of:

- The dynamic behavior (computations, communications, and migrations) of individual and cooperating agents;
- The basic mechanisms of the distributed architectures supporting agents, namely agent platforms;
- The functionality and emergent behaviors of applications and systems based on agents.

[3]Multi Agent System SIMulation framewOrk

The simulation is integrated to PASSI as a new process phase (as indicated in figure 11). If the simulation results are insufficient, new iteration is executed in order to make necessary changes. But, there is only one possible type of iteration which consists in returning to the Modeling phase. Although there is an integration of simulation in the process, however this phase can be seen as a test phase more than a simulation phase. We consider that what was proposed for PASSI do not correctly corresponds to a simulation-based design.

Sierra et al. ***[31]*** develop an Integrated Development Environment to design electronic institutions as multi-agent systems. The adopted process is depicted in figure 12. This process consists of:

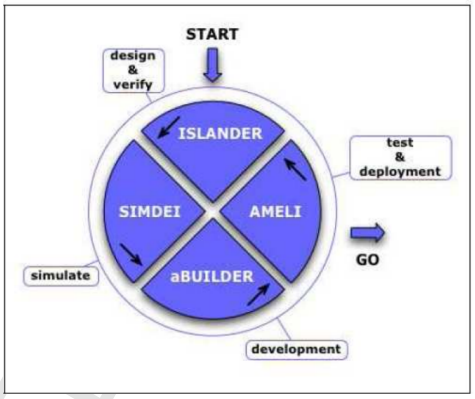


Figure 12: The Development Cycle proposed by Sierra et al. [31] for Electronic Institution.

- At the Design phase, electronic institutions can be graphically specified with ISLANDER which is a graphical tool that supports the specification of rules and protocols in an electronic institution. At the verification phase, the specified institution should be verified before opening it to external, participating agents.
- Then, a simulation tool SIMDEI is used to dynamically verify the specifications and the protocols to be implemented.
- At the Development phase, the verified institution can be implemented using the aBUILDER tool.
- Then, the electronic institution can be deployed.

With this approach, simulation is considered as a way to achieve what is defined as dynamic verification of the system. Simulations are realized using a simulation tool developed under Repast (which is multi-agent simulation tool). The analysis of the simulation results is the responsibility of the designer who can decide to return to the design phase if the simulation results differ from the expected ones. The problem with this approach is that the designer in not guided in the analysis task and the approach is not widely applied in various fields of applications. A wider range of application areas must be addressed in order to better evaluate it.

De Wolf et al. [15] uses an *equation-free* macroscopic analysis approach proposed in [25] and [24] to build self-organizing emergent systems. This approach consists in the definition of the macroscopic behavior (the agent behavior) using an equation (called a macroscopic equation) and numerical algorithms are used to obtain quantitative statements about the macroscopic properties. Traditionally, numerical analysis is applied to equation-based models. But, given that the complexity of the definition of a macroscopic equation for complex and dynamic systems, the equation-based model has been replaced by a realistic individual based simulation model.

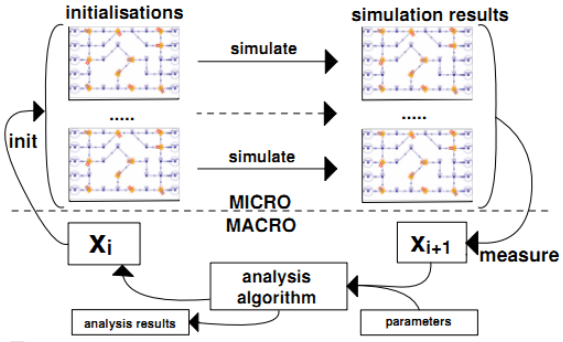


Figure 13: Equation-free accelerated simulation guided by the analysis algorithm proposed by De Wolf et al. [15] to build self-organizing emergent systems.

Figure 13 shows the adopted procedure to apply this approach which consists of:

- Initial values (X_i) for all the macroscopic variables under study are supplied to the analysis algorithm.

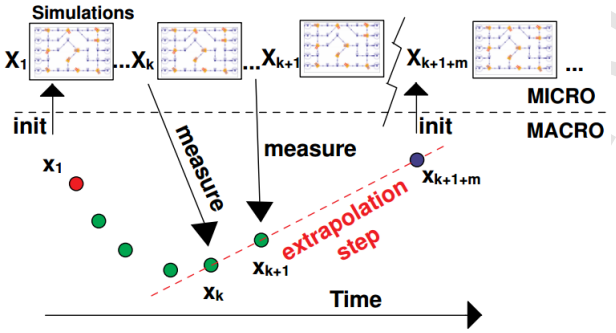


Figure 14: Equation-free accelerated simulation over time proposed by De Wolf et al. [15] to build self-organizing emergent systems.

- Then, the initialization operator (init) initializes a number of simulations according to these values.
- Next, simulations (simulate) are executed for a predetermined duration.
- Finally, the averages over all simulations of the values of the macroscopic variables (xi+1) are measured. These measures are then given to the analysis algorithm as a result. The equation and its evaluation are replaced by the simulation and the analysis algorithm processes the new results to obtain the next initial values.

This *init-simulate-measure* cycle is repeated until the analysis algorithm reaches its goal. Figure 14 shows this procedure over time. In this approach, designers have to define the results that are expected from the analysis, the parameters of the simulation are then initialized and simulations are launched. Finally, the simulation results are analysed and depending on the outcomes, the next initial values are determined. The analysis approach is integrated into the engineering process in order to achieve a systematic approach for building self-organizing emergent systems as the following:

- First, the prototype of the system is built based on experience and combining existing mechanisms and guidelines to achieve a self-organizing emergent system.
- Then the system is systematically analyzed with respect to the wanted macroscopic requirements using the analysis approach mentioned above.

- Feedback from that analysis is then used in a next engineering cycle to adjust and tune the solution in order to systematically evolve towards a final solution that meets all the requirements. The feedback obtained through the analysis of self-organizing emergent systems can also result in more experience and guidelines to use in future engineering processes.

The principal objective from proposing this approach is the systematic analysis of the agent behavior in order to decide if the required macroscopic behavior is achieved. The design is modified in an iterative manner until the system meets the requirements. The approach aim to achieve a systematic simulation-based engineering process where analysis and feedback are essential. However, this analysis approach is not exploited in an engineering process.

Gardelli et al. [16] proposes an approach to tackle the early-design stages in self-organizing multi-agent systems engineering. It is a three-stages design modeling, simulation and tuning approach. The approach is based on the A & A (Agent and Artefact) meta-model. This model describes a MAS in terms of agents and artifacts. Agents are defined as proactive goal-driven entities and artifacts are defined as encapsulating services to be exploited by agents. This approach consists of the following stages which can be executed in a cyclic way:

- Modeling: it is to develop an abstract formal specification of the system. The designer provides a characterization for user agents, artifacts and environmental agents. Environmental agents are used to embed self-organizing mechanisms into a MAS environment.

- Simulation: it is to use such a specification to qualitatively and quantitatively investigate the dynamics of the system.

- Tuning: it is to change model parameters and behavior so to adjust system dynamics. In the tuning phase, environmental agents behavior and working parameters are successively tuned until the desired dynamics are observed.

The objective from proposing this approach is not to develop a complete new methodology for MAS engineering. Instead, it is to integrate the approach within existing Agent Oriented Software Engineering methodologies, and addressing the peculiar issues raised by self-organizing MAS. They assume that

requirements have been collected and the analysis has been performed, in particular identifying the services to be performed by environmental agents. Then, this approach can be situated between the analysis and design phase.

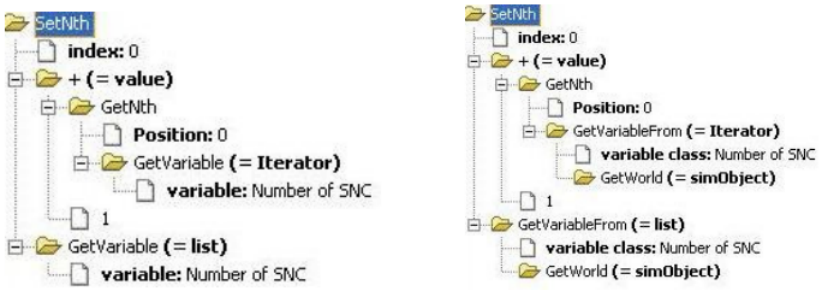


Figure 15: Actions Counting NCS under SeSAm. [8].

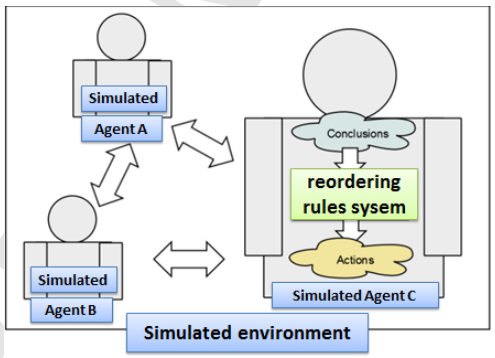


Figure 16: Location of the rules auto-scheduling system. [27].

Bernon et al. [8] focus on MAS based on the AMAS (Adaptive Multi-Agent System) theory in which the main design point is to improve the behavior of involved agents and removing the detected Non Cooperative Situations (NCS) (which represent problems of cooperation). They proposed an approach, based on simulation, to automatically identify NCS while a prototype of a targeted MAS is executing. The AMAS theory was proposed to help designers confronted with complex systems that are functionally adequate and for which classical algorithmic solutions do not exist. Figure 15 presents an example of NCS detection. Indeed, by observing the system

while it "lives", a designer can be aware of problems (NCS) that occur in the system and then solve them by (essentially) modifying the agents behaviors. This enables to design agents while the system is simulated. In this case, an agent can be partially designed and its capabilities of actions and reactions can be progressively improved by developers. A first simulation tool was-integrated in the implementation phase of the ADELFE process. This [7], based on the SeSAm platform [26], allows designers to see a MAS prototype during its execution in order to detect a number of NCS. The designer can then modify and improve the behavior of deficient agents.

Lemouzy et al. [27] define a tool enabling the self-scheduling of behavioral rules of a cooperative agent (figure 16) . It is an automatic mechanism of scheduling rules in order to select actions to be performed by an agent. A subsumption rule can be considered as a set of rules and each of them triggers one action. The modification of the deficient agents behavior steel, in these two previous works, done manually, but it would be interesting, to ease the task of the designer, to allow agents to self-define the set of behavioral rules which are considered the most cooperative.

Mefteh et al. [29] and [28], develop a new cooperative agent model which is able to self-design based on simulation. This agent model (named S-DLCAM [4]) gives to cooperative agent the ability to self-detect and self-correct the cooperation problems (Non Cooperative Situations) in order to self improve its behavior by anticipating the NCS during its life.

As a synthesis, several MAS methodologies have seen the day in recent years. MAS were heavily used to design and simulate many real problems. However, the use of simulation during the MAS's design process has been poorly applied. The principal objective of using simulation to design MAS is to better understand the behavior of each agent and to well identify the different interactions. Simulation enables to see the results given by a MAS model and if the designer isn't satisfied, he can rectify its model by changing the interactions or the agent properties. For self-organizing and emergent systems, simulation has been used because the designer has not a complete knowledge about the system behavior and its environment. In this case, sim-

[4]Self-Design and Learning Cooperative Agent Model

ulation has been used to complete the design. The designer gives generally a preliminary specification of the system and by using simulation, he can deduce new information which enables it to improve the design.

5. General Synthesis

Simulation has been used to design different types of complex systems. The objective is generally to eliminate unfit design as early as possible in order to decrease design cost. There are also some other specific objectives related to each domain of application.

The different approaches, using SBD for non-computer systems, are classified in two main classes: model-based simulation class and test-based simulation class. The model-based simulation consists in the simulation of an image of the system which is usually incomplete because the designer does not have complete knowledge about the system In this case, simulation gives the designer more information about the system and the domain and thus enables the possibility of completing the missing characteristics. The test-based simulation consists in the simulation of an image of the system. This image is usually almost complete because the designer can have complete knowledge about the system domain but he wants to simulate the functioning of the system before building the real one. We are not talking here about the difference between test and simulation but about two types of simulation. The test is another paradigm which differs from simulation. It consists in the experimentation of the finished real system in order to ensure its proper function.

The approaches using SBD for software are also classified in two main classes: approaches extending UML diagrams and other propose their specific process without reference to UML. Reusing UML paradigm consists on making extensions to UML diagrams and integrating simulation-based activities. Reusing standards and extending them is good in the sense that it favors the standardization of the new proposed methods based on these standards. But, it is not always possible.

For MAS, self-organizing systems and systems with emergent functionality, little development approaches have been proposed in the literature. And to systematically build a self-organizing emergent MAS remains an open issue.

This classification of related works is justified by the fact that my objec-

tive from carrying out this study is to inspire from existing related works to justify using a SBD approach for developing Adaptive Multi-Agent Systems (AMAS) and find the suitable way to integrate it in the ADELFE process. ADELFE is a methodology for designing software with emergent functionality and it is based on the AMAS theory. For this, I have decomposed works into three parts from the more general approaches to reach the closest to my domain. So, I first talked about using simulation to design non computer systems in several domains, then using simulation to design software and finally to design Multi-Agent Systems or systems that have one or more AMAS characteristics (self-Organizing Systems or Systems with Emergent Functionality).
AMAS [18] [20] systems are complex multi-agent systems with emergent functionality. Generally, the AMAS designer does not have a complete knowledge about the dynamic environment and he cannot deduce the complete global behavior of the system from building the agent behaviors and vice versa. Building AMAS systems can challenge even the most experienced designer. With applying SBD, we assist the AMAS designer by automatizing as much as possible its task and enhance the quality of the produced AMAS. Simulation plays a strategic role because it helps to support the modeling and simulation of agents that dynamically adapt their interactions, composition, and behavior. Indeed, SBD allows to realize a progressive AMAS design based on simulation by integrating a simulation-driven approach in the design process in order to enrich the ADELFE methodology by engineering activities and tools to help the designer and guide him to produce a better AMAS quality. The adopted simulation activities are depicted in figure 17.

- **A1: Define model.** The model is defined according to the thematic (analysis, design, ...) requirements. The output of this activity is a defined model.

- **A2: Preliminary validation.** The model is then preliminary validated according to properties which are required for the model. The result of this activity is the model to be simulated. This activity is done manually by an expert in order to eliminate errors that can be detected easily before launching simulations.

- **A3: Experimentation.** This is to experiment the preliminary defined model. This activity should be conducted by a tool which gives

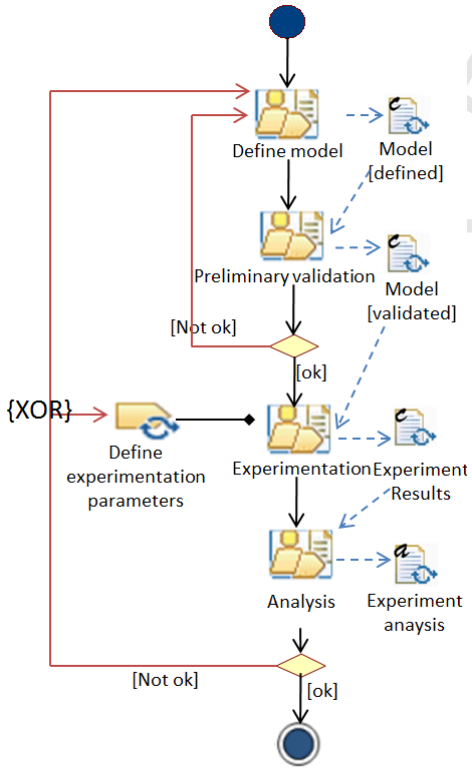


Figure 17: Our adopted Simulation-Based Work-flow Pattern

the ability to run experimentation by changing the experimentation parameters.

- **A4: Analysis.** This is to analyze the results produced in the previous activity and give conclusions. This activity also should be conducted by a tool.

If the conclusions produced by A4 satisfy the thematic, the process is stopped. Otherwise, a new cycle can be executed. The new cycle may begin either from A3 in order to change the experimentation parameters, or from A1 in order to refine the initial model.

References

[1] M Arno Louvo. Partitioning work centers for group technology: analytical extension and shop level simulation investigation. pages 267–290, 1992.

[2] M Arno Louvo. Casting Simulation as a Tool in Concurrent Engineering. Casting 1997-International ADI and Simulation Conference, Espoo, Finland. 1997.

[3] E Azarnasab, H Xiaolin, P Amini, and B Farhang. Progressive simulation-based design: A case study example on software defined radio. In *CASE*, pages 394–399, 2008.

[4] S Balsamo and M Marzolla. A simulation-based approach to software performance modeling. In *ESEC / SIGSOFT FSE*, pages 363–366, 2003.

[5] J Banks, S Buckley, S Jain, P Lendermann, and M Manivannan. Panel session: opportunites for simulation in supply chain management. In *Proceedings of the 2002 Winter Simulation Conference*, 2002.

[6] S Bansal. Supply chain planning: promise and problems of simulation technology in SCM domain. In *Winter Simulation Conference*, pages 1831–1837, 2002.

[7] C Bernon, V Chevrier, V Hilaire, and P Marrow. Applications of Self-Organising Multi-Agents Systems: An Initial Framework of Comparison. *Informatica*, 30(1):73–82, January 2006. ISSN 0350-5596.

[8] C Bernon, M-P Gleizes, and G Picard. Enhancing Self-Organising Emergent Systems Design with Simulation. In Gregory O'Hare, Michael O'Grady, Alessandro Ricci, and Oguz Dikenelli, editors, *International Workshop on Engineering Societies in the Agents World (ESAW), Dublin, 06/09/2006-08/09/2006*, volume 4457 of *LNCS*, pages 284–299. Springer-Verlag, 2007.

[9] P Byrne and C Heavey. Simulation, a Framework for Analysing SME Supply Chains. In *Winter Simulation Conference*, pages 1167–1175, 2004.

[10] Y Cahng and Hl Makatsoris. Supply Chain Modeling using Simulation. *International Journal of Simulation*, 2(1):24–30, 2001. ISSN 1473-804x.

[11] S Chin, S Maarten, and W Clancey. Multi-agent Modeling and Simulation Approach for Design and Analysis of MER Mission Operations. In *International Conference on Human-Computer Interface Advances for Modeling and Simulation (SIMCHI'05), Western Simulation Multiconference*, pages 73–78, 2005.

[12] D Cope, M Fayez, M Mollaghasemi, and A Kaylani. Supply chain simulation modeling made easy: an inoovative approach. In *Proceedings of the 2007 Winter Simulation Conference*, 2007.

[13] M Cossentino, G Fortino, A Garro, S Mascillaro, and W Russo. PASSIM: a simulation-based process for the development of multi-agent systems. *IJAOSE*, 2(2):132–170, 2008.

[14] J De Lara, E Guerra, A Boronat, R Heckel, and P Torrini. Domain-Specific Discrete Event Modelling and Simulation using Graph Transformation. pages pp. 1–30, 2012.

[15] T De Wolf, G Samaey, and T Holvoet. Engineering self-organising emergent systems with simulation-based scientific analysis. In S. Brueckner, G. Di Marzo Serugendo, D. Hales, and F. Zambonelli, editors, *Proceedings of the Third International Workshop on Engineering Self-Organising Applications*, pages 146–160, 2005.

[16] L Gardelli, M Viroli, M Casadei, and A Omicini. Designing self-organising environments with agents and artefacts: a simulation-driven approach. *IJAOSE*, 2(2):171–195, 2008.

[17] M Gaumann and A Sholapurwalla. Investment Casting Simulation. 2004.

[18] J-P Georgé, M-P Gleizes, , and P Glize. Conception de systèmes adaptatifs à fonctionnalité émergente: la théorie des amas. *Revue d'Intelligence Artificielle*, 17(4/2003):591–626, 2003.

[19] F Giancarlo, A Garro, and W Russo. A Discrete-Event Simulation Framework for the Validation of Agent-based and Multi-Agent Systems. In *WOA*, pages 75–84, 2005.

[20] M-P Gleizes, J-P Georgé, and P Glize. A Theory of Complex Adaptive Systems based on Co-operative Self-organisation. Demonstration in Electronic Commerce. 2000.

[21] R-G Ingalls. Introduction to simulation. In *Winter Simulation Conference*, pages 1379–1393, 2011.

[22] R Jayachitra and P Prasad. Design and selection of facility layout using simulation and design of experiments. *Indian Journal of Science and Technology*, 3(4):437–446, 2010. ISSN 0974- 6846.

[23] M Jensen. Simulation based design integration improves hybrid vehicle reliability. *Automotive Design Line Journal*, July 2006.

[24] I Kevrekidis, W Gear, and G Hummer. Equation-free: The computer-aided analysis of complex multiscale systems. *AIChE Journal. doi: 10.1002/aic.10106*, 50(7):1346–1355, 2004.

[25] I Kevrekidis, W Gear, J Hyman, P Kevrekidis, O Runborg, and C Theodoropoulos. Equation-free, coarse-grained multiscale computation: enabling microscopic simulators to perform system-level analysis. *Commun. Math. Science Journal*, 1(4):715–762, 2003.

[26] F Klügl, R Herrler, and C Oechslein. From Simulated to Real Environments: How to Use SeSAm for Software Development. In Michael Schillo, Matthias Klusch, Jörg P. Müller, and Huaglory Tianfield, editors, *MATES*, volume 2831 of *Lecture Notes in Computer Science*, pages 13–24. Springer, 2003.

[27] S Lemouzy. Auto-ordonnancement coopératif de règles comportementales. Master dissertation, University of Paul Sabatier, June 2007.

[28] W Mefteh, F Migeon, M-P Gleizes, and F Gargouri. S-DLCAM: A Self-Design and Learning Cooperative Agent Model for Adaptive Multi-Agent Systems. In *WETICE 2013 Conference, 11th Adaptive Computing (and Agents) for Enhanced Collaboration (ACEC), Hammamet, Tunisie, 17/06/2013-20/06/2013*. IEEE, 2013.

[29] W Mefteh, F Migeon, M-P Gleizes, and F Gargouri. Simulation Based Design for Adaptive Multi-Agent System: Extensions to ADELFE methodology. In *WETICE 2013 Conference, 11th Adaptive Computing*

(and Agents) for Enhanced Collaboration (ACEC), Hammamet, Tunisie, 17-20 Juin 2013. IEEE, 2013.

[30] B Shantanu and Y Narahari. Object oriented modeling and decision support for supply chains. *European Journal of Operational Research*, 153(3):704–726, 2004.

[31] C Sierra, J-A Rodraguez-Aguilar, P Noriega, M Esteva, and J-L Arcos. Engineering Multi-agent Systems as Electronic Institutions. 170, 2004.

[32] U Smutkupt and S Wimonkasame. Plant Layout Design with Simulation. In *Proceedings of the International MultiConference of Engineers and Computer Scientists*, volume 2, Hong Kong, March 2009.

[33] G Wainer and R Castro. DEMES: a Discrete-Event methodology for Modeling and simulation of Embedded Systems. *Modeling and Simulation Magazine. Society for Modeling and Simulation International*, 2011.

[34] H Xiaolinl. *A Simulation based design software development : methodology for distributed real time systems*. Ph.d. thesis, 2004.