Universidad Politécnica de Madrid
Escuela Técnica Superior de Ingenieros de Telecomunicación

PhD DISSERTATION

**CONTRIBUTION TO THE INTEGRATION OF
NETWORK SIMULATORS AND SOCIAL
SIMULATION ENVIRONMENTS FOR THE
MODELLING OF ENVIRONMENTS AND SMART
DEVICES**

Author

Álvaro Sánchez Picot

MSc, Telecommunications Engineer

Advisors

Tomás Robles Valladares

*Full professor in Departamento de Informática de Sistemas (DIS-UPM)
at Universidad Politécnica de Madrid (UPM)*

Diego Martín de Andrés

*Associate professor in Department of Telematics Engineering (DIT-
UPM) at Universidad Politécnica de Madrid (UPM)*

June 2019

Título de la Tesis: *CONTRIBUTION TO THE INTEGRATION OF NETWORK SIMULATORS AND SOCIAL SIMULATION ENVIRONMENTS FOR THE MODELLING OF ENVIRONMENTS AND SMART DEVICES*

Departamento: Departamento de Ingeniería de Sistemas Telemáticos

Autor: Álvaro Sánchez Picot

Directores: Tomás Enrique Robles Valladares, Diego Martín de Andrés

Tribunal nombrado por el Excmo. y Magfco. Sr. Rector de la Universidad Politécnica de Madrid, el día ____ de _____ de 2019

Presidente: _____

Vocal: _____

Vocal: _____

Vocal: _____

Secretario: _____

Suplente: _____

Suplente: _____

Realizado el acto de defensa y lectura de Tesis el día ____ de Junio de 2019 en la E.T.S. de Ingenieros de Telecomunicación de Madrid.

Calificación:

EL PRESIDENTE

LOS VOCALES

EL SECRETARIO

Abstract

This PhD Thesis work is developed in the areas of advanced communications, information technology and public safety by the National Institute of Standards and Technology (NIST) and is also embraced in the information society, a science area defined by the European Commission. The idea was conceived due to the lack of *Ambient Intelligence* (AmI) environment simulators that consider both, the people that traverse these environments and the devices existing in them. On the one hand, current existing social simulators such as *Multi-Agent Simulator Of Neighborhoods* (MASON), Repast or Swarm are focused on the people that interact in these environments but are not able to do an in-depth simulation of the devices and their communications. On the other hand, current existing network simulators such as ns-3, OMNet++ or NetSim are focused on simulating the devices in AmI environments and the communications among them but are not able to do an exhaustive simulation of people and their behavior.

In order to solve this problem, I propose in this work an AmI co-simulator called *Hydra*, that integrates MASON (a social simulator) and ns-3 (a network simulator). My proposal includes the necessary coordination mechanisms to make the whole simulation work, and all the modules necessary to perform the AmI simulation. Additionally, I propose a global data model, that integrates the objects from the social environment and the network environment; a sequence model able to integrate both simulations and a time model to calculate the limitations of the simulation. Based on the features required in a network simulator, a social simulator and additional ones identified in an AmI simulator, I proposed a methodology for the design of AmI simulators. I used that methodology to define the *Hydra* simulator.

Hydra was integrated into a real AmI environment obtaining the data from the sensors and actuators in the environment rather than generating them in the simulator. This integration enables the prediction of possible future events in the environment that could be prevented or at least minimize its effects. *Hydra* was also integrated with a semantic platform using a prosumer framework [Martin et al., 2013] for easy service creation by end users with no experience creating and deploying services over a semantic AmI platform. This framework facilitates the extraction of the data provided by the semantic platform, the execution of the simulations and the improvement of the user experience.

In order to validate the proposal, several experimentations, surveys and simulations were car-

ried out. The results from the validation concluded that: a more complete simulation of AmI environments can be obtained using *Hydra* than executing a social simulator and a network simulator separately, a methodology will help in the creation of co-simulators for AmI environments, future events can be predicted in an AmI environment using *Hydra* and prosumerization helps improving user experience.

In this work, I proposed the integration of ns-3 (a network simulator) and MASON (a social simulator) to create a system I have called *Hydra* and is based on a proposed methodology that coordinates both simulators. Data, sequence and time models were proposed to define *Hydra* and it was integrated into an AmI environment for the prediction of events and it was also integrated with a prosumer framework to facilitate its use. All this work was validated with positive results.

Resumen

Esta tesis doctoral se ha desarrollado en las áreas de comunicación avanzada, tecnologías de la información y seguridad pública según el NIST y también está incluido en el área científica de la información de la sociedad según la Comisión Europea. La idea del trabajo fue concebida por la falta de simuladores de entorno de Inteligencia ambiental que tengan en cuenta tanto la gente que atraviesa estos entornos como los dispositivos que existen en ellos. Por un lado, los actuales simuladores sociales como MASON, Repast o Swarm se centran en la gente que interactúa con estos entornos, pero no son capaces de profundizar en la simulación de los dispositivos y sus comunicaciones. Por otro lado, los actuales simuladores de red como el ns-3, OMNet++ o NetSim se centran en la simulación de los dispositivos de los entornos de Inteligencia Ambiental y en las comunicaciones entre ellos, pero no son capaces de hacer una simulación exhaustiva de la gente y su comportamiento.

Para resolver este problema, he propuesto en este trabajo de investigación un co-simulador de Inteligencia Ambiental que he llamado *Hydra* y que integra tanto MASON (un simulador social) como ns-3 (un simulador de red). Mi propuesta incluye los mecanismos necesarios de coordinación para hacer que toda la simulación funcione y todos los módulos necesarios para realizar la simulación de Inteligencia Ambiental. Adicionalmente, propongo un modelo de datos global que integra los objetos del entorno social y del entorno de red; un modelo de secuencia capaz de integrar ambas simulaciones y un modelo de tiempos para calcular las limitaciones de la simulación. Basado en las características necesarias en un simulador de red, un simulador social y otras identificadas en un simulador de inteligencia ambiental, he propuesto una metodología para el diseño de simuladores de inteligencia ambiental. He usado esa metodología para definir el simulador *Hydra*.

Hydra fue integrado en un entorno real de inteligencia ambiental obteniendo datos de sensores y actuadores en el entorno en vez de generarlos desde el propio simulador. Esta integración permite la predicción de posibles eventos futuros en el entorno cuyos efectos se podría prevenir o al menos minimizar. *Hydra* también se ha integrado con una plataforma semántica usando un sistema prosumer [Martin et al., 2013] para la creación sencilla de servicios por usuarios finales sin experiencia creando y desplegando servicios en una plataforma semántica. Este sistema facilita la extracción de datos que provienen de la plataforma semántica, la ejecución de simulaciones y la mejora de la experiencia de usuario.

Para validar la propuesta se han llevado a cabo experimentaciones, encuestas y simulaciones. Los resultados de la validación concluyeron que: se puede obtener una simulación más completa de entornos de inteligencia ambiental usando *Hydra* en vez de ejecutar un simulador social y uno de red de forma separada, una metodología ayudará a la creación de co-simuladores para entornos de inteligencia ambiental, se pueden predecir eventos futuros en un entorno de inteligencia ambiental usando *Hydra* y la prosumerización ayuda a mejorar la experiencia de usuario.

En este trabajo he propuesto la integración del ns-3 (un simulador de red) y MASON (un simulador social) para crear un sistema que he llamado *Hydra* y está basado en una propuesta metodológica que coordina ambos simuladores. Se han propuesto modelos de datos, secuencia y tiempo para definir *Hydra* y se ha integrado en un entorno de inteligencia ambiental para predecir eventos, así como también se ha integrado en un sistema prosumer para facilitar su uso. Todo este trabajo se validó con resultados positivos.

Acknowledgements

Dedication

A mi familia. Quien diría que iba a ser el primer doctor.

‘The oldest and strongest emotion of mankind is fear, and the oldest and strongest kind of fear is fear of the unknown.’

H. P. Lovecraft

Glossary

AmI *Ambient Intelligence*. 0.0, 1.1–1.4, 2.0–2.2, 2.4, 2.5, 3.0–3.2, 3.4, 4.0–4.2, 4.4, 4.7, 5.0, 5.1, 5.3, 6.0–6.2, 6.4, 7.0–7.6, 8.0, 8.2, 8.3, 9.0–9.3, 9.5, 10.0–10.4, 11.0

API Application Programming Interface. 5.2

BLE Bluetooth Low Energy. 1.1, 4.4

CORE Computing Research and Education. 11.0

CPSS Cyber-Physical Social Sensing. 0.0, 4.0, 4.1, 4.3, 4.4, 4.7, 5.0–5.3, 9.0, 9.2, 9.5, 10.2

CSS Cascading Style Sheets. 8.2

DSL Domain Specific Language. 4.4

FMI Functional Mock-up Interface. 4.1

GISAI "Grupo de Investigación de Servicios Avanzados de Internet". 9.1, 9.4

GUI Graphical User Interface. 4.6, 11.0

HLA High-level Architecture. 4.1

HTML Hypertext Markup Language. 8.2

ICT Information and Communication Technologies. 1.1

IoT Internet of Things. 1.1, 4.4, 8.1

IP Internet Protocol. 2.4

JCR Journal Citation Reports. 11.0

MASON *Multi-Agent Simulator Of Neighborhoods*. 0.0, 2.1, 2.4, 5.1–5.3, 6.3, 9.1, 9.2, 9.5, 10.1, 11.0

MASSIS Multi-agent system simulation of indoor scenarios. 9.2

MATLAB Matrix Laboratory. 5.2, 9.2

NetSim Network Simulator, and Emulator. 2.1

NIST National Institute of Standards and Technology. 0.0, 1.1, 1.4, 10.0

NS Network Simulator. 2.1, 3.0, 3.1, 3.3, 6.1–6.3

pcap Packet Capture. 2.4

PIR Passive Infrared Sensor. 7.2

RQ Research Question. 1.3, 3.0, 3.4, 4.0, 4.7, 5.0, 5.3, 6.0, 6.4, 7.0, 7.6, 8.0, 8.3, 9.0–9.2, 9.5, 10.0–10.4

SPARQL SPARQL Protocol and RDF Query Language. 8.2

SS Social Simulator. 2.1, 3.0, 3.1, 3.3, 6.1–6.3

TCP Transmission Control Protocol. 2.4

UDP User Datagram Protocol. 2.4

UPM "Universidad Politécnica de Madrid". 9.1, 9.2, 9.4

Wi-Fi Wireless Fidelity. 1.1, 2.4, 3.2, 4.4, 5.2, 7.5, 9.1–9.3

WiMAX Worldwide Interoperability for Microwave Access. 2.4

Contents

Abstract	i
Resumen	iii
Acknowledgements	v
1 Introduction	1
1.1 Context	1
1.2 Motivation	2
1.3 Hypothesis	3
1.4 Document structure	5
2 State of the art	7
2.1 Simulation	8
2.1.1 Social simulation	8
2.1.2 Network simulation	9
2.2 Ambient Intelligence	9
2.3 Service co-creation	11
2.4 Critical analysis of the State of the Art	12

2.4.1	Features	13
2.5	Conclusions	17
3	Proposal of an Ambient Intelligence simulator	19
3.1	Model features	20
3.2	Execution features	22
3.3	Analysis features	25
3.4	Conclusions	27
4	A Methodology for the Design of Application-Specific CPSS Co-Simulators	29
4.1	Methodology for the creation of AmI co-simulators	30
4.2	Final Users' Requirements and Characteristics Capture	33
4.3	Selection of the co-simulation paradigm	34
4.4	Particularization of the general simulation model and simulation life-cycle	37
4.4.1	Cyber-Physical Social Sensing Simulation Model	37
4.4.2	Simulation Life-cycle	38
4.5	Selection of the appropriate coordination mechanism	43
4.6	Design of the user interface and results presentation	44
4.7	Conclusions	45
5	Proposal of a Cyber-Physical Social Sensing Co-Simulator	47
5.1	Architecture	48
5.2	Modules	49
5.2.1	Experiment definition	49

5.2.2	Experiment control	51
5.2.3	Simulation Control	53
5.2.4	Data Manager	55
5.2.5	Data Analysis	56
5.2.6	Graphing tools	57
5.2.7	ns-3 Simulator	60
5.2.8	Mason Simulator	61
5.3	Conclusions	62
6	Models for an Ambient Intelligence co-simulator	63
6.1	Data model	64
6.2	Sequence model	66
6.3	Proposal of the integration of time models	70
6.3.1	Initialization of the simulation	71
6.3.2	Social simulator event	72
6.3.3	Network simulator event	73
6.3.4	Finalization of the simulation	74
6.3.5	Time aggregation	75
6.4	Conclusions	75
7	Integration of a simulator in an AmI environment to predict future events	77
7.1	Simulation integration scenarios	78
7.2	Data flow model for the integration of <i>Hydra</i> into the AmI environment	79
7.3	Decision-making and predictive solutions	84

7.4	Data processing	85
7.5	Assessment of the simulation model	86
7.6	Conclusions	89
8	Use of prosumerization for the simulation of AmI environments	91
8.1	Semantic platform	92
8.2	Prosumer framework	93
8.3	Conclusions	96
9	Validation	97
9.1	Proposal of an Ambient Intelligence simulator	98
9.1.1	Context	99
9.1.2	Plan	100
9.1.3	Data gathering	101
9.1.4	Results	102
9.2	A Methodology for the Design of Application-Specific Cyber-Physical Social Sensing (CPSS) Co-Simulators	106
9.2.1	Experimental validation: Co-Simulator Development and Experiment Description	107
9.2.2	Experimental Validation: Results	120
9.2.3	Discussion	128
9.3	Analysis of the integration of a simulator into an AmI environment	129
9.4	Use of prosumerization for the simulation of Ambient Intelligence environments .	132
9.4.1	Context	132

9.4.2	Planning	133
9.4.3	Data Collection	134
9.4.4	Result	134
9.5	Conclusions	139
10	Conclusions	141
10.1	<i>To obtain a more complete simulation of AmI environments using Hydra than executing a social simulator and a network simulator separately</i>	142
10.2	To propose a methodology that will help in the creation of co-simulators	143
10.3	<i>To predict future events in an AmI environment using Hydra</i>	144
10.4	<i>To improve the user experience with Hydra applying prosumerization techniques</i>	145
11	Future work	147
	Publications	149
	Bibliography	153

List of Tables

3.1	Simulators model features	21
3.2	Simulators execution features	23
3.3	Simulators analysis features	25
4.1	Coordination mechanisms.	39
4.2	Coordination mechanisms.	44
9.1	AmI simulator features fulfilled	103
9.2	Conclusions extracted by experts.	104
9.3	RQ3 evaluation summary.	105
9.4	Quality parameters results in Experiment#1	121
9.5	Scalability of the proposed co-simulators	127
9.6	Summary of the quality of the service built	135
9.7	Summary of the time spent in creating the services	136
11.1	Publications	154

List of Figures

1.1	Hypothesis, goals, research questions and experimentation	4
4.1	Implementation process of the methodology	32
4.2	Taxonomy of CPSS simulation paradigms	35
4.3	Base CPSS simulation model	38
4.4	Basis simulation life-cycle	39
5.1	Proposal of architecture for a CPSS co-simulator for AmI environments	48
5.2	Experiment Definition module workflow	50
5.3	Experiment Control module workflow	52
5.4	Simulation Control module workflow	54
5.5	Data Manager module interactions	55
5.6	Data Analysis module workflow	57
5.7	Graphing tools experiment selection screen	58
5.8	Graphing tools experiment configuration screen	59
5.9	Graphing tools experiment initialization screen	59
5.10	Graphing tools experiment execution screen	60
5.11	ns-3 Simulator architecture	60

5.12	MASON Simulator architecture	62
6.1	Data model of Hydra	64
6.2	Initialization of the simulation	67
6.3	New event in the social simulator	68
6.4	New event in the network simulator	69
6.5	End of the simulation	70
6.6	Simulation times equations	75
7.1	Simulator integration scenarios	79
7.2	Simulation data flow diagram	80
7.3	Simulation times equations	87
7.4	Simulation times requirements	88
8.1	Prosumer framework Architecture	93
8.2	Prosumer framework screenshot	95
9.1	Proposal of architecture for a CPSS co-simulator for AmI environments	98
9.2	Case study plan with inputs and outputs	101
9.3	Total of conclusions drawn	104
9.4	Overall evaluation about the utility of a co-simulator	105
9.5	Evaluation of the steps proposed by Hydra to perform a simulation.	106
9.6	CPSS Simulator Architecture	110
9.7	Simulator#1 architecture	111
9.8	Simulator#2 architecture	111

9.9	Basic configuration simulator interface	114
9.10	Simulation execution and control interface	114
9.11	Simulation design interface	114
9.12	Results presentation temporal interface	115
9.13	Simulator#1 architecture	116
9.14	Simulator#2 architecture	116
9.15	Simulation scenario	119
9.16	Distributions of services quality in Experiment#2	123
9.17	Distributions of services quality	124
9.18	Distributions of services quality	125
9.19	Distributions of services quality	126
9.20	Distributions of services quality	127
9.21	Graph with times from scenario 1	130
9.22	Graph with times from scenario 2	131
9.23	Distributions of services quality	135
9.24	Distributions of time spent in creating the services	137
9.25	Satisfaction evaluation	138

Chapter 1

Introduction

This chapter presents the context in which this thesis is included as well as the motivation that took me to do this work, introducing the current state in technology problems found, current tools that partially solve the problem and a proposal of a tool that integrates those systems. In addition, I present the hypothesis that supports this thesis, the goals associated with it the research questions that this work is going to address and the experimentation to support the answers to the research questions and thereby the hypothesis. Finally, I present the structure of this document with a brief description of each of the chapters.

1.1 Context

This thesis is embraced in the following areas from the NIST standard topics:

- **Advance Communications**, a topic mainly focused on wireless devices as it is estimated a great increase in the following years with more than 20 billion by 2020. This work focuses on simulation including the communication between devices and in Bluetooth Low Energy (BLE) and Wireless Fidelity (Wi-Fi) communications.
- **Information Technology** that also includes many subtopics such as networking, mobile data and informatics, all of them relevant in this thesis but with Internet of Things (IoT)

being the most relevant one due to the great increase in interconnections between devices and also a great influence in AmI environments.

- **Public Safety** as it is a great priority the safety of the people and one of the use case in this thesis is the evacuation of a building during an emergency situation.

In addition to the NIST topics, this thesis is also embraced in the **Information Society** science area by the European Commission that encompasses Information and Communication Technologies (ICT) where AmI environments, simulation and all this work is included.

1.2 Motivation

We are heading towards a technologically connected world where every day more and more devices are installed in our homes and our environment. Some of these devices are devices that have been in our lives for some time such as televisions, air conditioning units or security cameras but others are much more recent such as ambient lights, temperature sensors or microphones to talk with a computer. Currently, we want to know much more about what happens in our home and our surroundings than several years ago and thanks to the mobile phones we can easily access this information anywhere and in real time. Nowadays we also want the environment to act proactively depending on what happens, for example, turn on the lights automatically when the nightfall comes or turn them off when there is nobody present or open and close blinds depending on the light outside or the desired temperature inside the building. These environments where people and devices interact with each other, are called AmI environments.

The problem arises when one wants to deploy such an environment or improve an existing one, where the different devices try to cooperate together and take into account all the people that inhabit this environment in order to obtain the desired result. This is a complex task that needs to be carefully studied in advance and each case requires a specific solution. In addition, it is an expensive task, both in time and resources, that requires the installation of all the devices and the interconnections between them, so an error where some modification of the installation is needed could imply serious consequences again both in time and resources.

A tool that helps with this deployment, analyzing the environment and the interactions between the different elements in it, should simplify this task and reduce the chance of errors happening in the installation of the devices. Social simulators can be used to study the people behavior in AmI environments taking into account all the passive elements present there such as walls or doors but these simulators are not good simulating the behavior of the cyber-physical devices present in the environments such as sensors or actuators. On the other hand, network simulators can be used to study the behavior of the existing cyber-physical devices in AmI environments as these tools can simulate in great detail the behavior of sensors, actuators, routers or any other cyber-physical devices but they lack a good simulation of the people that interact and hang around in these scenarios, the other key element in AmI environments. The problem I have found is that currently there is no such tool that integrates both different aspects present in such environments: the social one, that focuses on the people moving around the environment, and the cyber-physical one, with all the different devices that can be found in these environments such as sensors, actuators or access points interacting with different protocols, technologies and standards.

1.3 Hypothesis

Based on all the problems previously stated in an AmI environment and the different elements that appear in this kind of environment, I state that if a system that unifies the people behavior in **social simulation** and the device and protocols analysis in **network simulation** is offered, the following goals can be achieved:

- **Goal 1:** *To obtain a more complete simulation of AmI environments using Hydra than executing a social simulator and a network simulator separately.*
- **Goal 2:** *To propose a methodology that will help in the creation of co-simulators for AmI environments.*
- **Goal 3:** *To predict future events in an AmI environment using Hydra.*

- **Goal 4:** *To improve the user experience with Hydra applying prosumerization techniques.*

In order to validate the previous hypothesis and goals, I propose the following Research Question (RQ):

- **RQ1:** Does the network and social co-simulation offer more features than the simple execution of each one separately?
- **RQ2:** Does the network and social co-simulation help in the analysis and drawn of conclusions?
- **RQ3:** What is the perception in the use of the co-simulator?
- **RQ4:** Does a methodology help in the creation of AmI co-simulators?
- **RQ5:** Does the integration of a simulator in an AmI environment help in the prediction of future events?
- **RQ6:** What is the quality of the services developed by prosumers?
- **RQ7:** Who are more efficient: prosumers or expert developers?

Figure 1.1: Hypothesis, goals, research questions and experimentation

Finally, in order to validate the previous hypothesis, goals and research questions I propose two experiments, one consisting in the execution of the simulation and another one in the deployment of an AmI environment where the data recollected will help with the comparison

against the data obtained in the simulation. Additionally, some surveys will be carried out to know the perception of the people using the co-simulator.

A sum up of the hypothesis, goals, research questions, experiments and the relation between them can be seen in figure 1.1.

1.4 Document structure

This document is structured as follows:

Chapter 1, *Introduction*, presents the context and motivation of this work and states the hypothesis, goals and research questions that will guide the thesis and will be a reference in all the future chapters.

Chapter 2, *State of the art*, studies all the related work to this thesis including simulation, social simulation, network simulation, ambient intelligence and prosumer.

Chapter 3, *Proposal of an Ambient Intelligence simulator*, analyzes the different features presented in social and network simulators and proposes an AmI co-simulator, called Hydra, composed of both simulators integrating their features but also generating new ones.

Chapter 4, *A Methodology for the Design of Application-Specific CPSS Co-Simulators*, presents a methodology to design co-simulators for AmI environments particularizing our co-simulator presented in chapter 3.

Chapter 5, *Proposal of a Cyber-Physical Social Sensing Co-Simulator*, presents the architecture and different modules of the Hydra simulator previously introduced in chapter 3.

Chapter 6, *Models for an Ambient Intelligence co-simulator*, presents the different models related to the Hydra simulator such as data models, sequence models and time models.

Chapter 7, *Integration of a simulator in an AmI environment to predict future events*, introduces the Hydra simulator in an AmI environment in order to predict future events that may

happen based on the data gathered from the sensors in the environment and use the actuators or some other elements to react accordingly.

Chapter 8, *Use of prosumerization for the simulation of AmI environments*, introduces a prosumer tool to the Hydra simulator to help non experts in the execution of the simulation.

Chapter 9, *Validation*, presents all the work done to validate the proposed tasks in the previous chapters.

Chapter 10, *Conclusions*, sums up all the work done in this thesis extracting the most important ones.

Chapter 11, *Future work*, suggests ideas about the improvement of the work presented in this text expanding it or applying it in different fields.

In this chapter, I have presented the context of this work related to different areas from the NIST standard and the European Commission, the motivation that took to the realization of it explaining the problems found and suggestions of solutions based on partial current solutions. This chapter also proposes the hypothesis that will govern this work in addition to the goals and research questions that support the hypothesis. Additionally, the experimentation carried out to validate all the work is proposed. Finally, in this chapter I also introduced the structure of this tome.

Chapter 2

State of the art

In this chapter I analyze the work related to the proposed contributions, linking the most important features with the objectives proposed in section 1.3.

The study of the state of the art focuses on **simulation** in section 2.1, particularly in social and network simulation. This work is related with all goals proposed but mainly Goal 1 ”*To obtain a more complete simulation of AmI environments using Hydra than executing a social simulator and a network simulator separately*” and Goal 2 ”*To propose a methodology that will help in the creation of co-simulators for AmI environments*”. Another important subject in this thesis and discussed in section 2.2 is **AmI**, related again with all goals but mainly Goal 1 ”*To obtain a more complete simulation of AmI environments using Hydra than executing a social simulator and a network simulator separately*” and Goal 3 ”*To predict future events in an AmI environment using Hydra*”. The final key subject discussed in section 2.3 is the **co-creation of services** related to Goal 4 ”*To improve the user experience with Hydra applying prosumerization techniques*”.

In addition to the subjects previously discussed, section 2.4 discusses the missing features found while analyzing the state of the art and then discusses the particular features of the social and network simulators used during this work that will lay the foundations for chapter 3 where those features and some more will be integrated in the co-simulator *Hydra* proposed in this work. All this work will be done taking into account the methodology for the design of co-simulators

proposed in chapter 4.

2.1 Simulation

Simulation is the process of designing a model of a real or imagined system and conducting experiments with that model and as such, it implies an approximation of real-world events [Sokolowski and Banks, 2011]. Simple simulations can be done with a mathematical model, but minimum complex ones require the execution of the simulation in a computer, as there are too many variables to take into account in a mathematical model. For this work I am going to focus on computer simulation, but it is still a broad field. It started as a scientific tool in meteorology and nuclear physics after World War II but since then lots of sciences have made extensive use of computer simulation, such as astrophysics, biology, economics, medicine or engineering [Winsberg, 2018].

There are many different types of simulations, each with a specific purpose [Law et al., 1991]. Related to AmI I am going to focus in two, the social simulation, that focuses on the behavior of the people and the network simulation that models the behavior of devices in a computer network.

2.1.1 Social simulation

Social simulation studies the interaction among social entities considering their psychology and their behavior [Davidsson, 2002]. The most extended types of social simulators are those based on agents where the person and the groups are modeled as agents. An agent is updated every certain time according to the behavior it has defined resulting usually in a new position for the agent and possibly influencing other agents.

There are different agent-based Social Simulator (SS)s such as MASON [mas, 2015], Repast [rep, 2018] or Swarm [swa, 2018], each with its own characteristics and usually particularized for a certain case study. Some of them work with a 2D environment, others have a 3D one

while others are text based simulators. The ones that include a graphic environment usually include some kind of physical engine to calculate the collisions between the agents and the environment.

A SS specializes in the behavior of the human and it can simulate other elements in an AmI environment such as sensors or actuators, but it won't be able to get an in deep simulation of those devices.

2.1.2 Network simulation

In a network simulation, a program models the behavior of a network and each entity present in it, as well as the messages sent between them [Breslau et al., 2000]. It can also simulate in detail the behavior of the entities such as routers or computers.

There are several simulators nowadays both open-source such as ns-3 [ns3, 2018] or OMNet++ [omn, 2017] and proprietary such as Riverbed modeler (former OPNET) [riv, 2018] or Network Simulator, and Emulator (NetSim) [net, 2018]. All are event driven, calculating the next event in the network such as the sending or receiving of a packet or the appearance of a new device within range and working accordingly [Siraj et al., 2012]. After the simulation they generate a log that contains all these events, useful for a future analysis.

A Network Simulator (NS) is very good at simulating the network in an AmI environment and can simulate the other elements in this environment, mainly the people, using specific algorithms for their movement but it can't do a very realistic simulation such a SS would, taking into account their behavior.

2.2 Ambient Intelligence

AmI is a discipline that makes our everyday environments sensitive to what is happening with the use of sensors, actuators, the network that interconnects all these devices and the server

that orchestrates all these elements [Cook et al., 2009]. The main objective of AmI is the improvement of people's life that use the environment. In order to achieve this objective the information generated in the sensors is recollected and processed in the server and then certain orders are sent to the actuators, based on the information gathered from the sensors [Sánchez-Picot et al., 2016]. The actuators in the end will influence the people that are present in the environment, ideally not being conscious of the technology [Sánchez-de Rivera et al., 2019] [Martín et al., 2016].

In AmI environments one can expect the following features [Cook et al., 2009]:

- **Sensitive:** The system needs to base its decisions on what is happening in the environment reacting to the people in them.
- **Adaptive:** The system also requires adjusting its behavior depending of the situation, considering the best possible behavior, and ideally anticipating to an event.
- **Transparent:** The people in an AmI environment should not be conscious of the technology that surrounds them. Thanks to the miniaturization of the technology this is easily achieved nowadays.
- **Ubiquitous:** An idea behind AmI is that it requires being present in as many places as possible, ideally everywhere. In this way there is more data recollected, and the more information, the best the system can react to a particular event.
- **Intelligent:** The system works using AI in order to achieve its goals. This is done recollecting the data from the sensors, processing it, and giving orders to the actuators in order to, in the end, influence the environment, specially the people.

AmI is mainly used in home environments controlling the elements of the house such as the air conditioning, the watering of the plants or the security but it can also be extended to larger places such as an office or a cinema [Bordel et al., 2019] to control those elements but also to prevent certain catastrophes such as a fire or, should it happen, manage the evacuation as best as possible guiding the people to the quickest and safer exit [Morales et al., 2014].

2.3 Service co-creation

Co-creation is a form of business strategy introduced by Prahalad and Ramaswamy that emphasizes the generation and ongoing realization of mutual firm–customer value [Prahalad and Ramaswamy, 2002]. Value, thus, is created through new forms of interaction and learning mechanisms used by firms and customers to share and combine resources and services.

Therefore, service provision by end-users offers an added value that is perceived by other users who are aware of the involvement of the former.

Resources, or goods, are seen as objects generated by services, whereas services are configuration of processes capable of differentiating and generating value for the stakeholders [Lusch and Vargo, 2014]. There is a clear shift from “good to services” in the provision of value, meaning that there is a shift from a static view based on single elements and/or relations to a dynamic view based on the service interaction process.

Vargo and Lusch proposed the Service-Dominant (S-D) Logic [Vargo and Lusch, 2004] to explain how a person who receives a service recognizes the value of the received service. S-D Logic determines service value as “the comparative appreciation of reciprocal skills or services that are exchanged to obtain utility”.

As the view of value depends on consumer’s nature and personality, they want to personalize his or her experience using a firm’s product-service proposition to a level which fits their expectations, and also allows the firm to obtain greater value from its investment in the form of new knowledge, more profit and customer loyalty. Regarding the latter, Pimentel and Reynolds [Pimentel and Reynolds, 2004] have shown that truly devoted consumers not only spread positive word of mouth but eventually engage in recruiting in order to convince or persuade others to get engaged in the co-creation project.

The Attention Investment model [Blackwell et al., 2009] offers a cost/benefit analysis of abstraction use that enables to predict the circumstances in which users will choose to engage in programming activities. According to this model, a user weighs four factors before taking an

action:

- Perceived cost: attention and monetary units to get the action done.
- Perceived investment: attention units expended toward a potential reward.
- Expected pay-off: reduced future cost, also measured in attention units, which will result from the action taken.
- Perceived risk: Probability of no pay-off or probability of additional future costs.

For example, a domain expert in a hospital pharmacy might be considering adopting a traceability system to detect drug transit. The expert might see a benefit (as an expected pay-off) in that this system would allow him to detect lost drugs and expired drugs in less time. The perceived cost is that he will have to spend time learning to use the new features, while the perceived risk is that the system won't be flexible enough to support the variety of business cases present in this environment. Some techniques [Miller and Myers, 2001] [Stylos et al., 2004] have explored these possibilities and define a trade-off point by which the project is viable from the customer's point of view.

2.4 Critical analysis of the State of the Art

After an in deep analysis of the state of the art I have detected several important features missing. In this section I am going to analyze the AmI environment to define the features that characterize a simulator for AmI environments. Because I understand that AmI environments integrate the social simulation with the simulation of the communications I am going to analyze the features of the AmI environments as a combination of social and network simulators with the objective of creating a prototype of AmI simulator using features from existing network simulators and social simulators complementing them with features none of them is able to offer.

For the co-simulator *MASON* [mas, 2015] has been chosen as the Social Simulator and *ns-3* [ns3, 2018] as the Network Simulator [Allan, 2010] [Siraj et al., 2012]. The reasons to choose these simulators over others are because they are free, widely extended, currently supported and I have experience working with both of them. The features presented in this section come mainly from these simulators.

2.4.1 Features

In this subsection I am going to list the different features of the actual social and network simulators analyzing what features does each simulator support [Luke et al., 2003] [Luke et al., 2004] [Luke, 2005] [Henderson et al., 2006] [Issariyakul and Hossain, 2011]. These features can be divided in model, execution and analysis.

I have identified the following **model features** associated with the social and network simulators. These features group everything related to the modeling of the different elements in a simulation such as people, devices or other elements in the environment; as well as other features that govern these elements or the environment such as the behavior, the mobility or the physics. All these features are related with everything prior to the execution of the simulation. Their explanation is as follows:

- **Environment element models:** Social simulators have different models for elements in an AmI environment such as walls or furniture. Network simulators can also model these elements and others that can affect the behavior of the devices such as temperature or humidity [Climent et al., 2013]. Network simulators also consider these elements in the signal propagation models.
- **Mobility models:** Social simulators have different mobility models based on the grid and usually for both the 2D and the 3D environments. Network simulators also have access to mobility models from the random movement to the predefined one.
- **People behavior models:** Social simulators have different models to how the people

behave in the environment. This behavior includes the movement and the interaction with other agents or with the environment.

- **Group behavior models:** Social simulators have models for the behavior of a group so that people's decisions are different when they are alone and when they are surrounded by others.
- **Human interaction models:** Social simulators have models that control the interaction between different people affecting their behavior.
- **Physics modeling:** Some social simulators need models to manage the physics in the environment in order to calculate collisions between people and between people and the environment. These might be useful in highly populated areas and when studying evacuations or other similar scenarios.
- **Network protocol models:** Network Simulators contain all different protocol models both wired such as Internet Protocol (IP), Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) and wireless such as Wi-Fi, Bluetooth or Worldwide Interoperability for Microwave Access (WiMAX). Social simulators have nothing related with network protocols and our AmI simulator, using the network simulator models also have those.
- **Network device models:** Network Simulator have models for common network devices such as routers but have the tools to create more complex devices using simpler modules.

I have identified the following **execution features** associated with the social and network simulators. These features contain everything related with the proper execution of the simulation such as synchronization, modifications or integration with other systems; as well as with its initialization such as the creation and configuration of the scene [Luke et al., 2003] [Luke et al., 2004] [Luke, 2005] [Henderson et al., 2006] [Issariyakul and Hossain, 2011]. Their explanation is as follows:

- **Preconfigured scenes:** The social simulators have different scenes preconfigured to work with such as for example an ant colony or a flock of birds in the *MASON*. Network simulators also have preconfigured scenes usually to check how different network protocols work.
- **Parameters configuration in scenes:** Social simulators enable certain variability in the scenes enabling the modification of specific parameters, without the need of recompiling the code, such as the number of elements in the simulation or a variable defining the movement of these elements. Similarly, Network simulators enable the modification of some variables such as the number of devices or the size of the packets sent.
- **Scene design tool:** The scene in a Social simulator is created programatically where the people and the environment are created via code. Similarly, all the devices and connections between them in a network simulator are programmed in the code. It is complicated to have a complete visual tool that can help in the creation of the scenarios because the elements and the relations can be very complex, but a basic tool should at least enable to draw the environment and place some of the elements in it.
- **Real-time execution:** Social simulators enable the execution in real-time being able to pause and resume the simulation or select the execution speed. Network simulators usually execute the whole simulation at once, but as they can be integrated with real-time systems, they can be modified to enable real-time execution and the possibility of pause and resume the simulation, but this is not usually a default behavior.
- **Batch execution:** In simulation one execution is not usually relevant but rather what matters is the execution of several simulations and the after analysis of the data generated. Social simulators enable the execution of batch simulations via command line with the possibility to modify some of the parameters during the execution. Similarly, Network simulators enable the execution of several simulations via command line each one generating its own report for a later analysis.
- **Real-time modification:** Social simulators enable the modification of some very specific parameters, but usually most of the parameters can only be modified before the beginning

of the simulation. As network simulators are not designed with the idea of a real-time execution, they don't enable by default the possibility of the modification of some of the parameters, but as they are integrated with real-time systems, they can be easily modified to enable these changes.

- **Integration with Real-time systems:** Social simulators are not usually designed with the integration of a real-time system. Otherwise, Network simulators, are designed with the possibility to be integrated in a real-time environment so that some of the data comes from real entities rather than from the simulation.
- **Discrete-event simulator:** Network simulators are discrete-event where each event represents something happening in the network such as a packet being sent, a disconnection of a device or a reboot of another. The execution of an event may create new future events (for example when a packet arrives to router it might be sent to another one). Social simulators are not discrete-event, but rather are synchronous.
- **Synchronous simulator:** Social simulators are synchronous where all the agents in the simulation are updated every certain amount of time (a second for example). Network simulators are asynchronous, they are discrete-event.

I have identified the following **analysis features** associated with the social and network simulators. These features correspond with all the data analysis the simulation has generated and can be seen in different ways such as visually or in text [Luke et al., 2003] [Luke et al., 2004] [Luke, 2005] [Henderson et al., 2006] [Issariyakul and Hossain, 2011]. Their explanation is as follows:

- **Real-time analysis:** Social simulators enable the possibility of an analysis in real-time: visually one gets access to the evolution of the position of different elements during the simulation and the inspection of these elements for further information. Network simulators enable a post simulation analysis of the data generated but not a real-time one.

- **Post execution analysis:** Social simulators enable a post execution analysis usually checking the logs generated during the simulation. Network simulators generate a file after the simulation ends that enable a post-visual analysis of the execution of the simulation.
- **Visual analysis:** Social simulators enable a visual analysis during the execution of the simulation but usually not afterwards. Network simulators enable a post execution visual analysis of the simulation.
- **Text logs:** Both Network and Social simulators enable the access of the logs generated during the simulation. *ns-3* can generate Packet Capture (pcap) trace files, that are files that contain the packages sent in the Network and can be analyzed using external network protocol analyzers such as *Wireshark* [Combs and Others, 2007].

In section 3.3 I use the analysis performed in this section as the foundations to the features proposed for our AmI simulator additionally identifying some that are missing in this analysis but that are important in that kind of simulators.

2.5 Conclusions

In this chapter I have analyzed the state of the art of technologies related to the goals of this PhD thesis: network simulation, social simulation, AmI and service co-creation identifying each of them with their related goals proposed in section 1.3. During this analysis I have identified existing features and performed an analysis of the different features found in social and network simulators that will be relevant in chapter 3.

Chapter 3

Proposal of an Ambient Intelligence simulator

Based on all the features presented in section 2.4, in this chapter I propose an AmI simulator I have called *Hydra* that integrates both a network and a social simulator and includes the features related to them in addition to some extra ones obtained thanks to the integration but that are not offered individually by any of the simulators. I am going to analyze the different features of the actual social and network simulators presented in chapter 2 comparing them to those in Hydra in order to see the differences. The work described in this chapter has been published here [Sánchez-Picot et al., 2018a].

As explained in chapter 2, I have divided the NS and SS features in three categories: model, execution and analysis features. Model features represent the models of the elements in the different simulators, this is the behavior and physics of the people in the social simulator and the networks and devices in the network simulator; they are explained in more detail in section 3.1. Execution features represent the characteristics necessary to configure the simulation and other extra features to improve the execution in addition to internal working of these simulations. These features are explained in more detail in section 3.2. Analysis features represent the possibility of processing the data generated during the simulation in order to gather knowledge from the simulation. Examples of these analysis are logs, visualizations or the possibility of

pausing the simulation. Analysis features are explained in more detail in section 3.3.

This chapter addresses **Goal 1**: *To obtain a more complete simulation of AmI environments using Hydra than executing a social simulator and a network simulator separately* presented in section 1.3 and answers RQ1: *Does the network and social co-simulation offer more features than the simple execution of each one separately?*. The experimentation that validates the contributions to this RQ is introduced in chapter 9. In the next chapter I will propose a methodology for the creation of co-simulators based on the analysis done in this chapter.

3.1 Model features

Hydra inherits the different models from both the network and the social simulators. Everything related to the people and their behavior is taken from the social simulator, this includes the individual behavior model, the group behavior model, the human behavior model and the physics model. On the other hand, everything related to the network is taken from the Network simulator, this includes the Network protocol model and the network device model. The environment model is inherited from both simulators while the mobility model, despite both simulators have it, is taken from the mason model because it is usually the people who will move around in the environment.

The table 3.1 summarizes the features supported by network simulator and social simulator based on the analysis performed in section 2.4 of the state of the art. In the last column of such table I identified the features of Hydra simulator, that as can be seen includes all the features from both simulators.

As can be seen *Hydra* features are not always present in both the SS and the NS, so it will be necessary to create this functionality in *Hydra* and deploy it depending on the support of the simulator one is working with. For example the devices are features existing in NSs but depending on the scenario people might need to interact with them so they also need to be added as an element in the SS but usually not all the characteristics will be relevant so a simplified version should be enough.

Table 3.1: Simulators model features

Model Feature	SS	NS	Hydra
Environment	✓	✓	✓
Mobility	✓	✓	✓
Individual behavior	✓	×	✓
Group behavior	✓	×	✓
Human interaction	✓	×	✓
Physics	✓	×	✓
Network protocol	×	✓	✓
Device	×	✓	✓

In order to make a proper simulator of an AmI environment the **environment** should be considered. A social simulator takes the environment in to account to calculate the pathfinding routes to move the people around and a network simulator also considers the environment to place the devices and calculate the propagation of the signal over walls and other obstacles. All this elements are necessary in an AmI environment and need to be included in Hydra. In addition, there are other environment elements that could be useful in a simulation such as ambient parameters (temperature, humidity, pressure...) that may affect a simulation.

In an AmI environment there are different elements that move around the environment and these movement patterns should be considered. A social simulation applies the **mobility models** to the people to move them around the scenario following certain rules such and a network simulator could also apply the mobility to some devices that may move around the environment such as drones. These mobility models consider the different elements in the environments such as walls, doors or other moving objects in order to calculate their routes avoiding collisions between them. Taking all this into account means that an AmI simulator needs to include the mobility in its models.

People in an AmI environment behave accordingly to their goals or certain patterns and this **individual behavior** is expressed in a social simulator where people with different behaviors can coexist in the same simulation. Examples of behavior could be the interaction of a person with elements in the environment such as doors or with devices such as actuators or the necessity to get to a certain point. This model needs to be included in Hydra.

People behave differently when they are alone than when in a group where for example the speed of the group usually adapts to the slowest member in their group. So there can be a personal goal as well as a group goal. The **group behavior** is important in a social simulator and should be taken into account in an AmI simulator.

The **human interaction** is a feature necessary in an AmI environment where people are one of the two key elements in the environment, the other being the devices. People interacting with other people can change in behavior or change their goal as one person convince the other one to do something different than he expected.

In order to get accurate environments, there needs to be some **physics model** to simulate the reality. Social simulators include these models so that the agents don't get through one another nor through elements in the environment. In AmI environments, where the people moving around can be a critical part to analyze, a simulator needs to take into consideration physics models.

Regarding the network simulator, an important aspect of AmI environments are the devices and in order to communicate between them and with other elements in the network there needs to be protocols models to use in the communications.

Hydra needs **device models** to represent the devices that exist in an AmI environment and interact with other elements in the environment and between them. These devices can be sensors that measure parameters such as temperature or humidity or expect some input from the user such as a button, can be actuators that could change the behavior of other elements in the simulation and can be network devices such as routers or a server.

3.2 Execution features

Hydra requires certain execution features that include the configuration of the scene and the proper simulation. Some of these features are included in either the social or the network simulator but in an AmI simulator it is required to aggregate them into a single tool as seen

in this chapter. Table 3.2 summarizes the execution features presented in chapter 2 including a column that represents their appearance in Hydra.

Table 3.2: Simulators execution features			
Execution Feature	SS	NS	Hydra
Preconfigured scenes	✓	✓	✓
Parameters configuration in scenes	✓	✓	✓
Scene design tool	×	×	✓
Real-time execution	✓	×	✓
Batch execution	✓	×	✓
Real-time modification	×	✓	✓
Integration with Real-time systems	×	✓	✓
Discrete-event simulator	×	✓	✓
Synchronous simulator	✓	×	×

Hydra has several different scenarios preconfigured. I define a scenario as an environment, usually composed of several rooms, the people inside this environment, their behavior, a possible special event that can happen such as the start of a fire and all the rest of the elements presented in an AmI environment such as sensors and actuators. The first preconfigured scenario is a very basic test with people with mobile devices moving around a router and connecting and disconnecting from the Wi-Fi; the second one is a section of a University with different sensors and actuators and explained in more detail in chapter 9. The scenes also have some parameters configurable to tune the simulation as required similarly to other simulators as explained in the previous subsection. It also enables real-time execution of a visual simulation with the capability to pause and resume and, once paused the elements of the simulation can be changed. There is also the possibility to run a batch simulation. The AmI simulator runs as a discrete-event simulator where the events are queued, even the periodic ones generated by the social simulator.

In order to simulate an AmI environment it needs to have some way to access **preconfigured scenes** that are going to be simulated and that serve as a reference to create the environment you need to deploy the devices and other elements in. This feature is included in Hydra having some preconfigured scenes and being able to create more if necessary and edit them as seen later.

For a simulation to output interesting results you need to be able to **configure certain pa-**

parameters in a scene so that you can test different variations and compare the different results. These parameters could include for example changing the number of people or devices in the simulation or guiding the people to try different evacuation places.

There needs to be a way to define and configure the environment you want to simulate selecting the key models you need in the simulation. An AmI simulator needs to integrate a **scene design tool** that enables this task, it can be as a visual tool one or editing a text file where the elements are defined following a schema.

A useful feature that can be included in an AmI simulator is the **real-time execution** of the simulation so that the user can visualize the evolution of the simulation and pause and resume as required. Hydra includes this feature enabling certain control over the execution of the simulation including the possibility of introducing certain elements and modifying the existing ones in the middle of a simulation.

When simulating AmI environments a single simulation is not enough to get useful conclusions [Ritter et al., 2011]. It is useful to get a general idea of the different things that are happening in the environment, but you need more data to extract some conclusions. Therefore a simulator should enable the possibility of executing several simulations at a time (what is called a **batch execution**) given certain parameters in order to get a massive amount of data of different outcomes, interesting in a post analysis.

While not being a key feature it could be useful that a simulator enables the possibility of **real-time modification** of the elements in the scene in order to try different configurations of the scene, check the reaction of the elements in the simulation to the changes and observe the possible results. One example where this feature can become important is when testing certain use cases such as the evacuation of a building where a new element (fire, earthquake or similar) is introduced in the middle of a simulation.

Simulators in a vacuum are a useful tool that enable the analysis of different outcomes for a specific environment as stated in chapter 2 but they can become an important element when **integrated with a real-time environment** where they monitor the state of the scene, can

even substitute real devices in the environment, get the data to improve the simulations or even to predict future events that may happen given the current status of the different elements.

Discrete-event simulator: In an AmI environment, events happen at different times during the simulation so the simulator should be **asynchronous**, but some elements might be synchronous such as the update of the elements in the scene, similarly to a social simulator where the state of the agents is updated every period of time.

3.3 Analysis features

The final kind of features included in a simulator are the analysis ones as discussed in chapter 2. These features are usually done by external tools once all the data of the simulation is gathered but the simulator can process some of this analysis. Table 3.3 represents a summary of these features and their existence in the different simulators as has been explained in section 2.4 in the state of the art. The last column is the existence of the feature in Hydra.

Table 3.3: Simulators analysis features

Analysis Feature	SS	NS	Hydra
Real-time analysis	✓	×	✓
Post execution analysis	✓	✓	✓
Visual analysis	✓	✓	✓
Text logs	✓	✓	✓
Batch data analysis	×	×	✓

Hydra enables the inspection of the different elements during the visual simulation in real time. An element in the screen can be selected and a overlay appears that shows information related to that element. Once a simulation ends a new screen appears that lets the user check the logs generated during the simulation enabling him to search and filter for a keyword. Also, as commented in the previous section, I am working on a tool to enable the analysis of multiple simulations at once.

As explained previously Hydra enables a real-time visualization of the simulation where the user can interact with the simulation in several ways. In addition to this some **real-time analysis**

can be also done where if an element of the simulation is selected the information related to it is updated as it evolves during the simulation. This includes basic information such as the position and some other more specific to the element such as their behavior or their goal.

Once a simulation is concluded there is usually a phase where the data generated during the simulation is analyzed, a **post execution analysis**. The most in depth analysis should be done by specialized tools such as MATLAB [mat, 2018] or R [err, 2018] but some basic one can be performed by Hydra that might be useful to get some first conclusions of the results in the simulation. This analysis includes the evolution of the position of the people during the simulation such as a heat map or graphs showing the historic values of a device.

After executing a simulation, a **Visual analysis** can help with the understanding of what has happened during the simulation and why it has finished in a certain state. Examples of this visual analysis can be in the form of graphs that tell the evolution of certain parameters during the simulation or a heat map showing the most visited positions in the environment.

Once a simulation finishes all the data generated during the simulation needs to be stored for a later analysis using the same tool but also so that an external tool can load the data and process it. These **data logs** need to be consistent with the type of data stored using a predefined schema that can be understood by these tools.

As commented previously executing only one simulation can be useful sometimes but it doesn't reveal important results. The final objective behind a simulation is to run several simulations (tens or hundreds) and then analyze the data generated to extract some relevant conclusions. This is where the possibility to perform **batch data analysis** becomes helpful. Some examples of this analysis is the possibility to aggregate the different values of an element in different simulations and be able to check the mean or median if appropriate as well as some getting some other information. Both NS and SS have the possibility to run batch simulations but require external tools to analyze the huge quantity of data generated.

3.4 Conclusions

In this chapter, I have analyzed the different features presented in a social simulator and in a network simulator in order to extract the ones necessary in an AmI environment and based on the analysis of those features done in chapter 2. These features have been classified in three different groups: model features that characterize the elements present in the social and network simulator, execution features that are those related to the proper simulation including the configuration necessary and the characteristics of the simulation, and finally the analysis features that are the ones related to the process of studying all the data generated during the simulation. With all these features I have proposed an AmI co-simulator called Hydra, that integrates both a social and a network simulator and as such it gathers features from both but it also gets new features originated from the integration of the simulators but not present in any of them.

In this chapter, I have addressed Goal 1: *To obtain a more complete simulation of AmI environments using Hydra than executing a social simulator and a network simulator separately*, presented in section 1.3 and answered RQ1: *Does the network and social co-simulation offer more features than the simple execution of each one separately?* The experimentation that validates the contributions to this RQ is introduced in chapter 9. In chapter 4 I will propose a methodology for the creation of co-simulators based on the analysis done in this chapter.

Chapter 4

A Methodology for the Design of Application-Specific Cyber-Physical Social Sensing Co-Simulators

In this chapter, a methodology for the design of CPSS co-simulators is proposed. This methodology contributes to the work of software architects, that conforms the initial requirement to the co-simulator creation. In this initial stage there is a conversation of the complexity of the simulations where several aspects are defined: system requirements, the users' profile, and the development team and the global lines of the future implementation. In later phases, at implementation time, more specific problems should be addressed, in specific crucial aspects about the data formats and time representation should be solved. I will delve into these details in chapter 6. The work described in this chapter has been published here [Sánchez et al., 2017].

In this chapter, I address Goal 2: *To propose a methodology that will help in the creation of co-simulators for AmI environments*, presented in section 1.3 and answer RQ4: *Does the network and social co-simulation offer more features than the simple execution of each one separately?* The experimentation that validates this RQ is introduced in chapter 9. In chapter 5 I will delve in the architecture proposed as a result of the methodology in this chapter and in chapter 6 I will analyze the models.

4.1 Methodology for the creation of AmI co-simulators

Interest has been reported in the research of co-simulation, that is, simultaneously and collaboratively simulating individual components each one with different systems [Bordel et al., 2018]. This enables the evaluation of deployments in a realistic way. To reach this, each simulation system needs to exchange different information such as data, needs to be synchronized in time and the whole system needs to orchestrate a coordinated simulation [Quadri et al., 2015]. There are two standard technologies for the coordination of co-simulations: the High-level Architecture (HLA) [Kuhl et al., 2000] and the Functional Mock-up Interface (FMI) [Blochwitz et al., 2011]. There are tools compatible with these standards but not all the simulation systems implement these standards as is the case in most social simulators. Additionally, tools that have integrated these standards have encountered certain problems and restrictions [Quadri et al., 2015]. Also these solutions do not deal with important challenges in relation to CPSS co-simulation (for example the FMI standard doesn't address time management nor data exchange). The truth is that the complexity of CPSS force designers to create their own application-specific solutions sacrificing the benefits of using standard solutions.

A four-step methodology for designing a CPSS co-simulator being able to fulfill the users' needs and requirements is proposed. The steps of this methodology will be explained in detail in the rest of the chapter:

1. Selection of the co-simulation paradigm.
2. Particularization of the general simulation model and simulation life-cycle.
3. Selection of the appropriate coordination mechanism.
4. Design of the user interface and results presentation.

Previously to these steps, I will need to determine: the requirements and characteristics applicable to the new co-simulation tool: "Final Users' Requirements and Characteristics Capture". From now on it will be the phase 0 of the methodology.

This four-step methodology should enable CPSS researchers and software architects to design the most proper co-simulation for a specific application, addressing this challenges [Palensky et al., 2014] [Quadri et al., 2015]:

- **Selection of the co-simulation paradigm** focuses on the “Data Flow and Concurrency”. This paradigm presents different approaches to parallelize and manage the data flow among the domain specific simulation tools and it provides different specifications to select among them based on the situation.
- **Particularization of the general simulation model and simulation life-cycle** focuses on challenges related to the “Modeling language” and “System Topology representation”. In this particularization it is proposed a basic simulation model, and instructions are provided to adapt it to the specific scenario. A similar process is performed with the simulation life cycle.
- **Selection of the appropriate coordination mechanism** focuses on the problems of “Time management” and “System Scalability”. This step presents several options to coordinate the different time representation and simulation speeds and additionally, it provides different criteria to select the most appropriate management technique depending on the desired scalability and the specific scenario.
- **Design of the user interface and results presentation** addresses problems of “Tool heterogeneity”. This step analyzes several ideas for the user interface in order to homogenize the interaction with the proposed co-simulator as much as possible.

As explained before there is a step 0 that requires an interaction process among software architects and final users. This proposal also includes and regulates this process and helps both, final users and designers, to determine the requirements and characteristics applicable to the new co-simulation tool. All the steps of this process can be seen in Figure 4.1 and are explained in this chapter. The implementation process, despite being the most time-consuming task, it is a post-design activity, so is not considered in this methodology proposal but will be explained later.

Figure 4.1: Implementation process of the methodology

4.2 Final Users' Requirements and Characteristics Capture

There has been a lot of studies discussing the requirements that co-simulators should satisfy in order to operate efficiently and correctly [Xu et al., 2017][Broman et al., 2015]. Despite this, most of these requirements may not be applicable or some other characteristics could be unconsidered in application-specific tools. Additionally, it might be complicated to interact with final users or might require a lot of time if a structured process is not adopted. Moreover, other than traditional requirements ought to be deemed such as final users' knowledge and the specific simulation type to be executed.

In this previous step, the specific requirements of the simulation tool and other characteristics should be defined by software designers and final users together. Four crucial thematic blocks need to be covered: technical requirements, application requirements and characteristics, development team characteristics and final user characteristics.

Related to **technical requirements**, they could be linked with one of the undermentioned properties [Zhou et al., 2012]:

- **Flexibility.** The adaptability of the proposal to other use cases, modes, environments, scenarios, etc.
- **Modularity.** The division of the co-simulator into as many modules as possible so these could be added or removed easily and without affecting other modules.
- **Scalability.** The maximum number of entities and the complexity of their behavior supported by the co-simulator in a specific scenario.
- **Accuracy.** The finer level of simulation usually the more real these will be, but it will also take more execution time to complete the simulation. This level of accuracy should be particularized for each use case as there might be scenarios where more detail is needed while in others the speed of the execution is important.

The flexibility and modularity properties are related to the programming structure of the co-simulator, specifically they represent the implemented level of opening in the co-simulator. Scalability entails that the co-simulator supports other possible scenarios and accuracy gives the means to the user to specify configuration parameters.

With respect to **application requirements**, they can also be linked with the earlier four technical properties identified: flexibility, modularity, scalability and accuracy. Lastly, **application scenario characteristics** such as the simulation scenario, **development team characteristics** such as their number and knowledge, and **final users' characteristics** such as their profile and skills, need to be debated among designers and final users to adapt the co-simulator tool to the specified context.

The first step of the proposed methodology, *Selection of the co-simulation paradigm*, and the third one, *Selection of the appropriate coordination mechanism*, consider co-simulators that fulfill the flexibility property. Co-simulator paradigms and coordination mechanisms are open solutions that take into account the modification of the co-simulator effortlessly, thus fulfilling the modularity property. The second step, *Particularization of the general simulation model and simulation life-cycle*, is designed to fulfill the modularity and scalability properties as the simulation model and life-cycle can be modified, extended or scale down independently following the users' requirements. Lastly, the fourth step, *Design of the user interface and results presentation*, fulfills the accuracy requirements using a user interface to configure the necessary parameters in the simulation. Despite this, all parameters could be affected by the four steps in the methodology.

4.3 Selection of the co-simulation paradigm

In step one, *Selection of the co-simulation paradigm*, after the definition of the specific requirements and characteristics for the simulation tool in step zero, the co-simulation paradigm needs to be selected. This section explains the different possibilities and the motivation to select the most reasonable paradigm based on the requirements defined in the previous phase.

Two different types of simulation schemes could be implemented when there is one than one domain in the scenario: independent simulation or co-simulation. The proposed taxonomy of the simulation paradigms applicable to the CPSS simulation can be seen in figure 4.2.

Figure 4.2: Taxonomy of CPSS simulation paradigms

In the **independent simulation** paradigm, each domain is simulated in an independent simulator specific for that domain. For this to work, the global scenario needs to be separated and configured into different subsystems by an expert, for example in the case of CPSS there are three subsystems: the physical, the cyber and the social ones. After the execution of the different simulators the results need to be combined to generate the results specific to the global scenario. There is a benefit in this paradigm, and it is the parallelization of the work as all subsystems are independent what greatly reduce the execution time, but the complexity in this paradigm is firstly in the division of the scenario and later in the merge of the results.

Co-simulation enables the processing of a description that includes all the features of the global scenario. They can be implemented with varying integration level what induces an increased simulation time. This solution requires a longer simulation time than an independent simulation.

There are two options for a co-simulation: integrated or federated. **Integrated co-simulation** are monolithic implementations where the features are not separated, and it works as a unique simulation algorithm. This makes it complex to create an integrated simulator right from the

scratch. Other works have stated the complexity of combining several tools and domains in a single simulation, even only two [Palensky et al., 2014].

On the other hand, in a federated co-simulation, several domain-specific tools are coordinated to simulate the whole scenario. In the case of CPSS there are three of these tools: a network simulator, a social simulator, and a physical process simulator. Each of these tools can be adapted to the necessities, but all the functional parts are separated between the tools. Depending on the modification required in each tool, two different paradigms can be recognized: orchestrated or choreographed.

Choreographed federated co-simulation requires the most level of development as all the different domain-specific tools need to be modified to simulate the global scenario, select the main data, configure them and exchange the information between the other domain-specific tools. In order to achieve this modification a certain level of programming is necessary. In this paradigm one of the simulators generates the final simulation results based on the data from all the simulators. The choreographed federated co-simulation is the most similar to the integrated simulation previously discussed.

In **orchestrated federated co-simulation**, it is not necessary the modification of all the domain-specific simulators, so no programming knowledge is necessary because an entity processes the global scenario description and configures the simulation for each specific tool based on that description. Once the simulation ends, this entity also generates the final simulation results based on the data from the tools. There are two different options for this entity in an orchestrated federated co-simulation. It can be **a third-party engine** where no modification in the domain-specific simulators is necessary because this engine uses the simulator interfaces to communicate with them, or it can be **one of the simulators** so in this case it would require some modifications.

The selection of the co-simulation paradigm is a complex task that depends mainly on the technological knowledge of the future developers and future users, and the features of the selected tool that will compose the co-simulator.

4.4 Particularization of the general simulation model and simulation life-cycle

After selecting the most suitable co-simulation paradigm in the previous step, now it is necessary to consider the specific application the simulator is being designed for. For this step the characteristics and final users' requirements discussed in step zero should now be considered as inputs 4.2, particularly the application use cases and the type of simulations. The application scenarios will allow designers the description of a specific simulation and the type of simulation could help in the proposal of a proper simulation life-cycle. All these is described in this section.

4.4.1 Cyber-Physical Social Sensing Simulation Model

The simulation model on the designed co-simulator should be centered on the most important feature of the reality for the specific chosen application. Characteristics from the different subsystems could be present, but the description ought to be much more thorough when representing specific aspects of the world.

A basic simulation model for CPSS can be seen in figure 4.3 which could serve as the foundations for an own model. This model is later explained in detail in chapter 6.1. There are four categories of the elements in the model: general elements common in the three worlds (physic, social and cyber), elements from the physical world, devices particular to the cyber world and those belonging to the social world.

General elements cover the limitations and the behavior of the agents in the simulation. **Social elements** represent mainly the people, the relations among them and their state. **Cyber elements** include devices such as sensors and actuators. The rest of are part of the **physical world elements**.

Based on the proposed model in figure 4.3, the most proper simulation model should be developed considering the specific application if was designed for. In order to accomplish this, some aspects of the model could be reduced, for example ignoring the distinction between sensors

Figure 4.3: Base CPSS simulation model

and actuators, but other aspects could be extended, for example including more details in the definition of a person.

There has been some proposal of methodologies for the creation of the right and complete simulation models [Sargent, 2013] [Jeruchim et al., 2006]. According to the purpose of the application and the designed co-simulator, some aspects could be ignored [Duarte et al., 2006]. Some global guides are introduced in table 4.1 to build the simulation model based on the selected application, the scenarios, and specific problem that the co-simulator needs to solve.

4.4.2 Simulation Life-cycle

The simulation life-cycle could differ based on the kind of simulations performed for example adding more steps. These steps are going to affect mainly step four in the methodology, *Design of the user interface and results presentation*. As an example, if all the simulations executed attend to a single goal, there is no need for "goal specification" step and an interface is not necessary.

The design of the life-cycle is easier than defining the simulation model. A general simulation life-cycle can be seen in figure 4.4 that can be used as a base to define the specific life-cycle

Table 4.1: Coordination mechanisms.

Validation of AmI systems	It is necessary a definition of the physical or ambient world. Specifically, models for the evolution of important phenomena such as the spreading of a fire. Additionally, it should be considered the addition of different kinds of people taking into account their age, special needs, etc.
Development of IoT systems	The concept of “service” should be included in the simulation model, and it could be necessary a more detailed description of the different kinds of devices. For example, a description of the interfaces.
Social research	For this case most of the particularization of elements in the cyber world can be ignored, being enough the concept of device. Additionally, all elements related to the social world need an extension such as for example including distinct states or interrelations.
Crowd management	In this case it could be necessary to detail the physical object including models for walls, doors or other elements in the building in order to manage the people as properly as possible. Additionally, it could be interesting the addition of emotions and their propagation in the crowds.
Social sensing	Specific sensors to this use case need to be considered such as presence or cameras. Additionally, the social world needs a more in-depth description similarly to the one described in “social research” or “crowd management”.

adding or removing steps when required. If the tasks corresponding with a specific state are not necessary that state can be removed, but if new tasks have to be included then new states have to be defined.

Figure 4.4: Basis simulation life-cycle

Co-simulation environments are generally used to test the interaction among humans, devices and other infrastructure objects. Simulations that show certain characteristics of these relations are executed and refined as much as needed to obtain information about users and devices. The

quality analysis of all the steps in the life-cycle [Cioffi-Revilla, 2014] is described below:

Goal specification: During this step stakeholders establish the problem to solve, using requirement acquisition techniques such as field studies or surveys and then the research problem is formally formulated. In the case of simulations that integrate social behavior and device interactions a goal usually pursued is to monitor the operation of a control system, comparing its results with the execution context. This control system is a device or set of devices that controls the behavior of other systems in the smart environment. This is a step previous to the definition of the simulation model associated to the system, and the goals can be utilized to model what experiments are necessary for the specific problem and situation.

Definition of simulation model: Social behavior and social interactions with devices and the control system need to be integrated into the simulation model. Some considered models seen previously in figure 4.3) are person, their behavior both individual and collective, devices such as sensors or actuators and environmental elements.

Domain Specific Language (DSL) are used to define simulation models by model-driven development techniques. Different end-user development tools existing in the co-simulation framework will be used to transform these models to simulation specifications. These tools can be reused in different domains and enable the developer or any other stakeholder the description of specific characteristics of the problem.

Experiment definition: Simulation experiments consider the selected models based on the characteristics to be covered. It is recommended to consider the agent behavior, and all the important elements included in the developed simulation model. Simulations that consider at least one of these characteristics will be executed and evaluated by the developers. The necessary simulations will be defined by the experiments model and will incorporate certain information, such as the building layout or possible limitations in the devices.

The experiment definition is performed using configuration tools. Specific applications are selected, and the desired models are selected and combined such as population, situation, behavior or devices. Experts would enable end-users the job of creating the experiments reusing

information and letting them decide the relevant information that ought to be extracted from the simulation for a later analysis. The result of the application of these tools are the simulation specifications.

Simulation execution in co-simulation system: In this step the simulated problem is parameterized resulting in reusable components particular to the problem that are required by the co-simulation framework. The actions from the experiment definition and control system behavior are used on the simulation in order to affect the actors and to produce the desired effect. The system should be adjusted to manage the varying situations that could happen such as unexpected problems in devices. Additionally, the communication among devices is tested considering its effectiveness, cost and potential problems such as bandwidth or data quality.

Simulation verification: This step is necessary to confirm that the simulation model and the experiments were properly defined. In this step both passive and active tests are performed to validate the model confirming that it behaves as expected. Additionally, some tools and techniques should be considered like unit testing, code debugging or profiling [Sargent, 2013] and test suites such as spatial analysis, histograms or pattern matching. These techniques enable as the validation of the proposed model and the implementation of the framework. But some questions need to be answered: Considering the definition and implementation of social behavior, is the selected social theory properly instantiated using adequate tools or programming languages? Is the implementation of the social behavior computationally efficient? Does the publication and obtention of user behavior models have an architectural implication?

Results analysis: In this step the results from the experiments are analyzed based on the domain knowledge in order to improve the current model and identify possible new requirements. A successful experiment generates a detailed documentation about the expected behavior of the system. This work can then be used to follow the behavior in a real environment.

Considering the deployment process in the CPSS environment of services or control systems, it can be assumed that the system behavior in a simulation would be like the one in a real environment. Utilizing a software-in-the-loop approach, one can be sure that the conditions of these services are similar enough to the reality.

After the deployment, the information obtained during the experiments can be utilized to decide if the actual observation in a real AmI environment corresponds to a simulated scenario or if it is required the execution of a simulation with the actual context in the real world to detect differences related to the experimentation and/or to predict evolutions based on the actual situation.

But some questions need to be answered: Considering the effectiveness, does the model supply what is required to answer the proposed research questions? Is the quality of the simulated infrastructure that supply the most effective simulation experience good enough?

Finally, to properly follow the defined simulation life-cycle, it is necessary to consider the subsequent requirements, based on those explained in section 2.4:

- **R1:** In CPSS, it is necessary to consider characteristics of the agents' behavior related to the social, physical and cyber worlds. As an example, the movement of a device such as a smartphone can be modeled by the movement of the person carrying it. Additionally in social simulation it can be useful to know the Wi-Fi coverage in a room or if a BLE beacon detects a broadcasting signal, because people behavior could be influenced by those or other events.
- **R2:** Certain tools are necessary to define experiments based on previous ones changing functionalities or stimulus that affects the simulated agents in order to produce the expected result. The creation of the experiment ought to be done by domain experts because they have the necessary experience to detect human and device behavior in knowledge bases but usually do not have experience in software engineering. Due to this situation it is necessary to provide tools that enable configuration and personalization adapted to the domain experts' skills.
- **R3:** The execution of a simulation generates a huge amount of data, so tools that helps in the analysis of the results are very important. These tools ought to be able to process and ease the expert's analysis. Additionally, once the services or control systems are deployed, these tools ought to be used to analyze and compare the data that is being

generated with the simulations in order to identify possible deviations and predict future situations. In order to achieve this, the simulator must generate performance data.

4.5 Selection of the appropriate coordination mechanism

Some of the proposed co-simulation paradigms don't need additional decisions because it defines the whole structure of the new co-simulator. Problems with the data format or the communications protocols in the coordination mechanisms needs to be selected in general designs. If the selected option is a choreographed paradigm, then all simulators need to be fully compatible so there is no need to change the data format nor the protocols because all simulators implement the same. On the other hand, orchestrated co-simulation needs the orchestrator to apply all the elements such as protocols and interfaces used by the other tools, so again no change is necessary in the data format nor in the protocols. In federated co-simulation, the main aspect is the coordination among the different domain-specific tools that compose the global co-simulator. In this case only decisions about the synchronization mechanism needs to be taken that is the focus of step three in the proposed methodology. Particularly two coordination mechanisms ought to be implemented, parallel execution and stop and wait as can be seen in table 4.2.

The most important step in the proposed methodology is the definition of the coordination mechanism, despite it might seem simple following the guidelines in table 4.2, the codification of these strategies implies several challenges. Some examples are the ordering of the different events from several domain-specific simulators or the unification of the time representation when each simulator uses a different data format or accuracy. The solutions to these problems are specific to the co-simulation paradigm chosen in step one of the methodologies but at least an initial synchronization between the simulators needs to be done to homogenize the time. After this, the simulation can be executed reevaluating the state after each event or time step.

To choose the best coordination mechanism two aspects need to be considered: how hard it is to implement the mechanism and the simulation time necessary to execute a simulation with that specific mechanism. The last aspect is related with the scalability requirements defined

Table 4.2: Coordination mechanisms.

Parallel execution	All	With this mechanism each tool is processed separately in another host, processor or thread. Calculations are executed in parallel and the results are shared among the other simulators as soon as available managed by the orchestrator element if required.
Stops and waits execution	Choreographed	In a predefined order each simulator processes its calculations and when the last one finishes the results are shared with the other simulators.
	Orchestrated by a third-party engine	In a predefined order the engine commands all the simulators to start their calculations. As soon as each simulator ends, it sends the results back to the engine that resends the information to the other ones.
	Orchestrated by one of the simulators	The orchestrator starts executing its calculations and shares the results with the other tools. After that, in a predefined order it commands each other simulator to execute its calculations. When simulator ends it sends the results back to the orchestrator that then resends it to the other tools.

in step zero 4.2. As a general rule, implementing a parallel execution is much expensive than a stop and wait mechanism but the time for a simulation to execute is less when considering parallel execution. The analysis of these temporal aspects is analyzed for a specific case in section 6.1.

4.6 Design of the user interface and results presentation

Step four of the methodology, the last one, is centered on the design of the user interface personalized to a specific application, simulation model, and life-cycle and centered in the presentation of the simulation results. This step is very influenced by the other steps, as an example, if the "goal specification" state is not present in the simulation life-cycle, then there is no necessity to develop an interface that shows the information. So, the interface will adapt their components to display only the important aspects of the simulation based on the decisions in other steps. In addition, the accuracy requirements defined in step zero need to be considered as well as the final users' profile to adapt the environments to their necessities.

The user interface of any simulator needs to include at least these three functionalities [Cao, 2002]: mechanisms to design a simulation, to execute and control the simulation and to analyze the results. For each of these ones there are some decision that need to be considered:

Mechanisms to design a simulation. The user interface needs to allow the selection of a scenario between all the possible ones defined in the simulation process, including modeling and configuring the different elements found in a specific simulation. To do this, there needs to be a way to design the scenario including the adequate rules for a right scheme. There are two possibilities: the simulations use predefined layouts or external tools are linked to the co-simulator to define a new scenario for each simulation.

Mechanisms to execute and control the simulation. There needs to be a way to interact with the simulation that could be different depending on the execution, the objective and the number of simulations to be executed. The main options to consider are either a Graphical User Interface (GUI) or a text interface.

Tools to analyze the results. There needs to be a way for the user to analyze the simulation. There are two possible situations: the results can be presented "live" so the user can see the way the simulation evolves and maybe even interact with it pausing it or the results can be presented "post-mortem" once the simulation has ended the results are shown to the user either in a visual way, with graphics and images or with text.

4.7 Conclusions

In this chapter, I have proposed a methodology for the design of application-specific CPSS co-simulators consisting of several steps. The first step is the selection of the co-simulation paradigm that is going to be used. The second step is the particularization of the general simulation model and simulation life-cycle. Then the third step is the selection of the appropriate coordination mechanism and finally, the fourth step is the design of the user interface and results presentation. Before these steps a requirements and characteristics capture is taken.

In this chapter, I have addressed Goal 2: *To propose a methodology that will help in the creation of co-simulators for AmI environments* presented in section 1.3 and answered RQ4: *Does the network and social co-simulation offer more features than the simple execution of each one separately?*. The experimentation that validates the contributions to this RQ is introduced in chapter 9. In chapter 5 I will delve in the architecture proposed as a result of the methodology in this chapter and in chapter 6 I will analyze the models.

Chapter 5

Proposal of a Cyber-Physical Social Sensing Co-Simulator

In this chapter, I present a proposal of architecture for a CPSS co-simulator based on the analysis done with the social simulators and network simulators in chapter 3 and the methodology proposed in chapter 4.

In section 5.1, I analyze the architecture in overall with the integration of the different simulators and the different modules that are integrated that are explained in detail in section 5.2 divided in those necessary for the definition of the experiment, then those related to the execution of the simulation and finally I will analyze in the detail both simulators.

In chapter 6, I will continue the design of the architecture proposed in this chapter with the definition of data model, sequence model and time analysis.

This chapter addresses Goal 1: *To obtain a more complete simulation of AmI environments using Hydra than executing a social simulator and a network simulator separately* defining the architecture of the simulator that will later be used in the experimentation in chapter 9 to validate the contributions to RQ1, RQ2 and RQ3.

5.1 Architecture

Based on the analysis made on the previous chapter, I propose in this section an architecture for a CPSS co-simulator for AmI environments integrating existing functionalities of network simulators such ns-3, as well as social simulators such as MASON, with additional components for integrating and uniforming the behavior of the final system.

The first required step for simulation is to define the experiment. **In Hydra an experiment is defined as different models and objects that are later particularized into a simulation.** This definition is later explained in detailed in section 5.2. This includes models of different objects that populate the simulation such as people, devices or furniture as explained later in chapter 6; the topography of the scenario where the experiment takes place; events that can take place during the simulation such as a fire, an earthquake or a blackout; and ending condition events that mark the finalization of the simulation, these events can be a time or a condition such as that all the people get to a certain region.

Figure 5.1: Proposal of architecture for a CPSS co-simulator for AmI environments

In figure 5.1 it can be seen in detail the architecture proposed. I differentiate 3 main elements in the architecture. First there is the top part of the architecture, where one can find the modules that enable the user the creation, particularization and execution of an experiment including all the elements described before, as well as the subsequent analysis of the simulation. The other 2 main elements in the architecture are ns-3, that will manage the network simulation and MASON that will manage the social simulation. All these modules and their interactions are explained in detail in the next section.

5.2 Modules

In this section I present the different modules that compose the CPSS co-simulator as presented in the previous section. This modules are the Experiment Definition where the user will select the models that define an experiment: the Experiment Control that will enable the user the particularization of an experiment for its subsequent simulation; the Simulation Control that will manage the simulation execution coordinating both simulators; the Data Manager that manages all the models, experiments and other data generated, storing and retrieving it as necessary; the Data Analysis that offer the user some tools to analyze the data generated during the execution of a simulation; as well as the ns-3 network simulator that will manage the cyber-physical devices during the simulation and the MASON social simulator that will manage the people and their behavior during the simulation.

5.2.1 Experiment definition

This module manages the definition of an experiment for a later particularization and simulation in the Experiment control and Simulation control modules.

In Hydra an experiment is defined as different models and objects that are later particularized into a simulation. So an experiment is composed of the following characteristics:

- A set of object models such as people or devices

- A set of communication models used between the devices
- A topography of the scenario including the map and the layout of the different static elements such as walls or doors, events models such as a blackout or a fire, and the ending conditions.

All these characteristics of an experiment can be defined in the Experiment definition module by an expert. A workflow of the process can be seen in figure 5.2. The processes that appear in the workflow can be done in any order but when programmed one needs to be defined. When all the processes are completed the experiment definition is created and stored for later access in the experiment control module.

Figure 5.2: Experiment Definition module workflow

The first task of the Experiment definition module is a query to the database to extract the different models stored and previously defined and presenting them to the expert that is going to define the experiment. Then the expert needs to select the different models that are relevant

to the experiment he wants to create. Examples of these models include different behaviors of the people, different sensors or actuators, as well as communication protocols such as Bluetooth or Wi-Fi.

Then the expert needs to input the layout of the map where the experiment is going to take place indicating the different static elements that are relevant in the simulation so that the agents can move and behave according to them.

After that, the expert can define the models of the events that can happen during the simulation and that are key to the simulation in order to check the behavior of the different elements before and after these events. The occurrence of these events is optional and can include a fire, a blackout, an earthquake or a crowded space.

Finally, the expert defines one or several ending conditions for the simulation based on the models stored. These conditions can include the length of the simulation, people that need to reach an area or a value detected in a device.

All these elements consist an experiment definition, the particularization of the elements in the experiment and the addition of extra information consist an experiment and are done in the Experiment Control module.

All the experiment definition is then stored in the database so that different experiments with different parameters, but the same models can be simulated later.

5.2.2 Experiment control

The Experiment Control module manages the creation of an experiment based on an Experiment Definition previously created. The Experiment Definition is particularized for each model defined with information received from the user that may include the number of different elements of a particular model, their position in the map as well as the connection between the different devices in the experiment. In addition to particularize the models defined in an experiment definition, more information is added to complete the definition of the experiment. This

can include the number of simulations that are going to be executed. Figure 5.3 represents the steps necessary to particularize an experiment to be used later by the simulation control module as explained in section 5.2.3.

Figure 5.3: Experiment Control module workflow

The first task of this module is the extraction of the different experiment definitions stored in the database so that the user can choose the one he wants to particularize and execute.

Then, the user needs to introduce several parameters for each model, these includes both parameters specific to the model such as the period of updating of a sensor and, if appropriate, the quantity and the place of these objects in the scenario.

In addition to these models the user needs to input the number of simulations that are going to be executed, and if there is only one, he can choose to control the simulation visually.

Finally, the user needs to specify information related to the ending conditions of the simulation if any.

Once all this information is defined the simulation can start, but the control of each simulation,

as there can be more than one, is managed by the Simulation Control module.

5.2.3 Simulation Control

The Simulation Control module manages the simulation during its execution orchestrating both simulators as the simulation progresses. There are three differentiated blocks in the process of the simulation control. The first one is the initialization of the simulation where each simulator gets the data necessary to start the simulation. The second one is the execution of the simulation where the simulation advances alternating between the network and the social simulator as an algorithm predicts. The last block is the finalization of the simulation where all the data generated during the simulation is sent to the Experiment Control module that will store it in the database once the data from all the simulations is gathered. Figure 5.4 represents a workflow of the necessary steps to control the simulation where the different blocks can be seen.

First the Simulation Control module receives all the data from a particularized experiment from the Experiment Control module. This data is then decomposed between the social simulator and the network simulator so that each simulator receives only the information that is relevant to their simulation. In a Network Simulation most of the data related to the people is not necessary, such as the behavior, only the position is interesting, similarly in a Social Simulation the protocols the devices use to communicate are not relevant. The decomposed data is sent to each simulator so that they can initialize their own instance of the simulation.

During the execution of the simulation the module communicates with both simulators alternating the execution as the simulation advances. This process is explained in detail in section 6.2. This module decomposes the information received from one simulator to the other after each update similarly as done when the initialization of the simulation.

This module analyzes the data generated during the simulation to check for several conditions. It checks if an event has happened as defined in the experiment. This condition can be for example a time after the simulation has started. If an event triggers the necessary information

Figure 5.4: Simulation Control module workflow

from the models is sent to each simulation changing the course of the simulation as necessary. Similarly, this module checks if an ending condition has been met, in case affirmative the module starts the process of ending the simulation as explained in the next paragraph.

Finally when the simulation ends the corresponding data is gathered from each simulator and all the data generated during the simulation, from both simulators and from this module is sent to the Experiment Control module that will later store the data for a future analysis.

5.2.4 Data Manager

The data manager module controls the the data stored and retrieved from the data base. It manages the data retrieved so that it corresponds to what is being searched but its main function is to adapt the data received and store it in an appropriate format. In figure 5.5 it can be seen all the different modules that interact with Data Manager as explained later.

Figure 5.5: Data Manager module interactions

This module processes the data received from different sources in order to adapt it to the format required in each case. Similarly, it receives data from different sources that require to be stored in the database and this module adapts this data so that all the data stored in the database has the same format.

All the modules that work with data, use the Data Manager module in order to consult what data is available.

The Experiment Definition module that needs the different models that are stored in the database uses the interface from the Data Manager to consult what models are available, and similarly when the experiment definition is created it sends it to the Data Manager so that it stores it.

The Experiment Control module ask the Data Manager module what experiment definitions are available so that the user selects the one he wants to experiment with. Once the experiment is complete all the data generated during it is then sent to the Data Manager in order to store it.

Finally the Data Analysis module needs the data from the experiments that have been done and gets all the data from the Data Manager module. Any conclusion from the analysis will be sent back later, once the analysis has finished, to the Data Manager module.

5.2.5 Data Analysis

The data analysis module enables the user to analyze an already finished experiment to try extract relevant information from the data gathered during the simulations in the experiment. A workflow of this process can be seen in figure 5.6.

This module asks the Data Manager module for all the experiments that have been simulated so far, so that the user can choose the one he wants to analyze. Once one is selected the Data Manager sends all the data generated once the simulations of the experiment finished including the data generated during the social simulation, during the network simulation and events that may have happened or other information in the Simulation Control module.

Once the experiment is selected the user can select a specific simulation (if there are more than one) or analyze the data of the experiment calculating different parameters. If one simulation is selected it can be analyzed visually in detail with the options of pausing, rewinding and fast

Figure 5.6: Data Analysis module workflow

forwarding the simulation or the data can be analyzed separately.

In addition to the visual analysis of a simulation, the Data Analysis module enable the visualization of different graphs gathered from the data of the experiment. These graphs are related to the parameters of the simulation and any can be selected to observe their evolution in time. An example of the graph could be the data from a presence sensor. Also, 2D maps that represent the historical position of the people in the simulation are available, what could be considered as a heat map of the simulation.

All the information generated during the data analysis can be downloaded to keep it for the future but no information is stored in the database, but the experiments remain stored so more future analysis can be done.

5.2.6 Graphing tools

The user interacts with the simulator using *Graphing tools* that enable him to use the different modules in order to create, particularize and execute an experiment as commented in the

previous modules of Hydra.

The user is first presented with a list of the different experiments that are available. These experiments have been previously defined by the Experiment Definition module and stored by the Data Manager. An image of this screen can be seen in Figure 5.7. When the user selects one experiment then the *Graphing tools* accesses the *Experiment Control* module in order to present the user with the different parameters and information related to the selected experiment. Some of the parameters are common to all the experiments such as the number of simulations to do, but some others are specific to the models selected during the *Experiment Definition* such as the number of certain devices or the frequency of information. An image of this screen can be seen in Figure 5.8. Once the parameters have been filled there are two options, the user can select a visual simulation or a batch simulation without graphic representation. In both cases the *Simulation Control* is accessed, in the first case the user has control over the flow of the simulation, while in the second he only receives the results of the simulations. An image of the simulation starting can be seen in Figure 5.9 and an image of the simulation being executed can be seen in Figure 5.10.

Figure 5.7: Graphing tools experiment selection screen

The Graphing tools also enable the user the analysis of an experiment previously done. In this case the Data Manager is consulted and a list with the different experiments executed are presented, the user can select one of them and then a different tool appears that enables the analysis of the data generated during the experiment. Alternatively all the data can be exported to be analyzed with an external tool such as Matrix Laboratory (MATLAB) [mat, 2018] or R [err, 2018].

HYDRA SIMULATOR [Home](#) [About](#) [Contact](#)

Configuration

MASON parameters

MASON URL

NS3 parameters

NS3 URL

Scene parameters

Initial people

Random Multiplier

Tstep(ms)

Batch simulation parameters

Number of Steps

Number of Simulations

Figure 5.8: Graphing tools experiment configuration screen

Figure 5.9: Graphing tools experiment initialization screen

Figure 5.10: Graphing tools experiment execution screen

5.2.7 ns-3 Simulator

This section presents the architecture of the ns-3 simulator with an Application Programming Interface (API) to control the simulation with messages from the *Simulation Control* module. This architecture can be seen in Figure 5.11.

Figure 5.11: ns-3 Simulator architecture

Once the experiment is defined by the Experiment Definition module and then particularized by

the Experiment Control module in the simulation, this last module sends the ns-3 the necessary information to start its own network simulation with the parameters specified by the user and the models defined in the experiment.

While ns-3 is prepared to run simulations on its own and run the simulations without pausing, I have slightly modified the basic architecture in order to be able to pause, resume and update a simulation as necessary by the Experiment Control module in the simulator using certain messages defined in an API. These messages enable the beginning of the simulation adding the necessary data to perform that action, the continue of a paused simulation being able to add new data if necessary, like the updates of the position of the people in the simulation calculated by the social simulator and finally the finishing of the simulation.

ns-3 has its own persistent storage where it can store the simulation once executed in order to later analyze it, and there are also some tools associated with the simulator that enable the analysis and visualization of a simulation. Both these elements are not used while working with the ns-3 inside the simulator as these tasks are managed by other modules in the simulator as explained in this chapter.

During the simulation the ns-3 updates the simulation as it is informed by the Experiment Control module.

5.2.8 Mason Simulator

This section presents the architecture of the MASON simulator with an API to control the simulation with messages from the *Simulation Control* module. This architecture can be seen in Figure 5.12.

Once the experiment is defined by the Experiment Definition module and then particularized by the Experiment Control module in the simulation, this last module sends the mason the necessary information to start its own social simulation with the parameters specified by the user and the models defined in the experiment.

Figure 5.12: MASON Simulator architecture

5.3 Conclusions

In this chapter, I have proposed an architecture for a CPSS co-simulator for AmI environments using ns-3 simulator as a network simulator and MASON as a social simulator. This proposal is based on the analysis of the simulators done in chapter 3 and the methodology for co-simulators proposed in chapter 4.

I have also analyzed the different modules that compose the architecture, first all the elements that enable the user to define an experiment based on different models, particularize an experiment using the parameters defined by the user and then execute the simulations of the experiment; and finally the ns-3 simulator and the MASON simulator.

In this chapter I have addressed Goal 1: *To obtain a more complete simulation of AmI environments using Hydra than executing a social simulator and a network simulator separately*, defining the architecture of the simulator that will later be used in chapter 9 in the experimentation to validate the contributions to RQ1, RQ2 and RQ3.

Chapter 6

Models for an Ambient Intelligence co-simulator

With the proposal of the architecture of the AmI simulator in chapter 5 based on the analysis done in chapter 3 and the methodology in chapter 4, in this chapter I propose models for data, sequence and time for the co-simulator Hydra. Specifically in section 6.1 I present the data model that represents the different elements that can be found in an AmI environment related to both the social simulation and the network simulation, in section 6.2 I define the sequence model in detail, in particular the messages sent between the *engine* and the simulators to control the simulation in the different stages of the simulation. Finally, in section 6.3 I do a proposal of integration of time models for Hydra where I measure the time each different task takes during a simulation.

This chapter addresses Goal 1: *To obtain a more complete simulation of AmI environments using Hydra than executing a social simulator and a network simulator separately*, presenting the models used in the simulator that will be used in chapter 9 in the experimentation to validate the contributions to RQ1, RQ2 and RQ3.

6.1 Data model

In this section, I define the data model corresponding to the elements in an AmI environment as can be seen in figure 6.1. This model is used by the *experiment definition module*, the *experiment control module*, the *simulation control module* and the *data manager module* as explained in section 5.2.

Figure 6.1: Data model of Hydra

This model is a general conceptualization of the different elements one can find in an AmI environment and are used in the NS, in the SS or in both. This is a general model valid for both simulators but the Engine manages the particularization of the model for each simulator as needed because not all the elements are relevant to both simulators as will be seen in each case.

I have divided the object class in two different general categories. On the one hand, there is the agent that corresponds to everything that has certain intelligence and might take decisions one way or another as will be seen. On the other hand, there is the environmental element that correspond to all the elements that don't have an intelligence, that don't take decisions based on their relationship with other elements as well as other environmental information.

The **environmental element** corresponds to elements found in AmI environments and is divided in two categories. The first category of the physical object includes all inert objects

found in the defined space. Considering a closed space such as that of a building, one can find in room elements such as walls, doors and windows and inside the room one can also find different types of furniture, columns and some other elements. These inert objects are especially interesting for the SS because the people are going to take them into account in order to traverse the environment but cyber-physical elements such as communication lines or cables can also be found.

In addition, the second category inside environmental element is ambient variable and as its name implies here one can find everything related to the ambience such as temperature, humidity or luminosity, information that can be relevant in some cases to the simulation as for example a low or high temperature could influence in the behavior of the people inside the room or even in the performance of the electronic elements in a device. All the elements in this category, environmental elements, can be used by both the SS and the NS, and as explained before, the physical objects could be interesting for the SS to calculate the pathfinding the agents should do but these elements could also be used by the NS when calculating the attenuation of the propagation of a wireless signal.

Then there is the **agent** that represent all that requires certain intelligence in the simulation in opposition to the elemental environment. It includes two subsequent groups. One of the groups is the people that inhabit the environment and this part of the model contains all the information relevant in the social simulation such as their relationship with others or with other elements what will make them behave in a way or another. The other group is the device and this group is going to be relevant in the network simulation, this includes elements such as sensors, actuators, drones or routers as well as computers including servers that might be gathering the data that is generated in the network. As already said the person is going to be a relevant part of the model in the SS and the NS will only be interested about the position of the people. Similarly, the device is going to be a relevant part of the model in the NS and the SS will only be interested in their position and in the actuators to interact with them. The *engine* will take care of this and only share with each simulator what is relevant to each one as explained in the next subsection.

In addition to the two categories that the agents are divided into, there is a **behavior** that controls all these elements and this can include information such as the movement of the agent or the interaction with other agents or with the environment.

All objects might possess certain relevant information such as its size, position, weight or the material they have which might be interesting for the NS to check how the wireless communications propagate. All these details are stored in the **physical model**.

In figure 6.1 there is a particularization of the general model for the SS. The SS requires mainly detailed information in everything related to the people, while not very exhaust information about cyber-physical devices, only their movement and the possible interaction between them and people.

In figure 6.1 there is also a particularization of the general model for the NS. The NS requires mainly detailed information in everything related to the cyber-physical devices, while not exhaustive information about people, only their movement and the possible interferences in the signals. The NS might also require detailed information about the material and the density of the elements in order to calculate the signal propagation through those elements.

6.2 Sequence model

Based on the *Simulation control module* proposed in section 5.2.3 I identify four possible scenarios in an AmI simulation and for each one I have defined a sequence model explained in detail in this section. In order to explain these scenarios, it is necessary to explain first how the *Simulation control module* works with the information from both simulators.

The *Simulation control module* (section 5.2.3) keeps track of the future events that are going to be processed in either the NS or the SS and correspond to the different simulations. These events are stored in an Event Queue and are extracted by order, always getting the first one in time that corresponds to the next simulation. The NS generates the future events and knows when they are going to happen, so for example if a packet is sent a new event is generated that

corresponds to the time when the packet will arrive at the other side of the communication line. The time of this event is sent to the *Simulation control module* and is stored in order in the Event Queue. The SS updates all the agents each step in a initially defined time. The events corresponding to this time, called *Simulation control module*, are stored in the Event Queue.

Once the simulation starts there are four possible situations, the first one is the proper initialization, the second one are the events from the NS, the third are the events from the SS and the last one is the simulation conclusion where the information generated is processed and stored.

The first scenario is the initialization of the simulation and how the model is particularized for each simulator and all the relevant information that appears. In figure 6.2 how the data flows between the different components can be seen.

Figure 6.2: Initialization of the simulation

The first step of this scenario is when a user initializes the simulation and then the *Simulation control module* starts generating the initial data necessary for the simulation. This data includes everything related to the scenery including the position and dimension of the different elements such as the people, routers and cyber-physical devices as well as walls doors and windows. This information can be randomly generated or previously defined depending on the configuration the user has specified. One of the most important parameters of the configuration is the

$Tstep$ parameter defined as the delay between each simulation and used mainly for the social simulation as this works updating the position of all the agents each $Tstep$.

Once the information has been generated it is then particularized for each simulator as defined in the models because not both simulators need all the information nor all the details. The data is then sent and each simulator starts the simulation using the information. Once done each simulator informs the *Simulation control module* and once both simulators have ended the simulation is ready to begin.

Both simulators halt the simulation and only update it when informed by the Engine. Once initialized there are two possible cases until the simulation ends, in the first the social simulator is the one that needs to run the next simulation while in the other is the network simulator the one that continues with the simulation.

Figure 6.3 shows how the synchronization is done when the next step in the simulation corresponds to the SS and how this information is sent to the NS should it need some of the data generated by the SS in its following steps. This corresponds to the second scenario.

Figure 6.3: New event in the social simulator

In order for this to happen the event buffer that keeps the Engine checks that the first event (which also corresponds to the first event in time), belongs to the SS. It then removes this event

from the queue and informs the SS to continue with the next step. The SS then updates the agents generating certain information that could be relevant to the other simulator. The Engine processes this information and sends it to the NS that updates its information and informs the Engine once finished.

Figure 6.4: New event in the network simulator

Figure 6.4 shows how the synchronization is done when the next step in the simulation corresponds to the NS and how this information is sent to the SS should it need some of the data in its following steps. This corresponds to the third possible scenario.

In order for this to happen the Engine removes the first event in the queue and detects that it corresponds to a NS event. It then informs the NS which then executes the next event. Once done it has generated new information that is sent to the Engine. Between this information there could be something relevant to the SS, so the Engine proceeds to extract this information and send it to the SS that will update its agents according. The information generated in the NS could also contain new events that are added to the Event Queue.

Finally, in figure 6.5, how the finalization of the simulation is executed can be seen, this is the last scenario. This happens when the Engine Event Queue is empty, when the simulation time has expired or when there are any other event that may cause the simulation to end such as the user finishing it manually.

Figure 6.5: End of the simulation

In the finalization, firstly the Engine informs both the SS and the NS to end the simulation. They then proceed to process all the information generated during the simulation and send it back to the Engine. Once the *engine* has the data from both simulators it then processes it and stores the information in a database so that the user can easily access it in a future time. Once the information is stored, the simulation ends and the user is informed.

6.3 Proposal of the integration of time models

Based on the analysis of the sequence model of the simulator done in the previous section, in this section I analyze the time needed to process the simulation. In this way the limitations can be known, what could be useful, as we will see in later chapters, if for example one wants to integrate the co-simulator in a real environment. The work described in this section has been published here [Sánchez-Picot et al., 2018b].

As seen in section 6.2 there are four different communication phases between the simulators and the *engine* and these are: Initialization, SS event, NS event and ending. In this chapter I will analyze in detail those messages and the time needed since the process is started until it ends.

6.3.1 Initialization of the simulation

During the process of the initialization of the simulation there are different steps necessary to complete.

First the *engine* needs to access the database in order to retrieve the data related to the simulation scenario and the models of the different agents involved in the simulation (people, cybernetic devices and their behavior) as well as the models of the environment objects (walls, doors, windows, furniture and communication lines). I name this time t_{idb} .

Second, the data obtained from the database is then decomposed for each simulator (t_{ien}). The SS requires data related to the agents and their behavior but not in deep details of the communication used in the scenario. Otherwise the NS requires all the data relevant in the communication such as wireless technologies, protocols or propagation delays but only requires the position of the agents, not their behavior. All this information is particularized with the parameters introduced by the user as explained in chapter 5.

Finally the decomposed data is sent to the SS (t_{tss}) and the NS (t_{tns}), to each its corresponding data. The SS will process the data and start its own simulation with the data received (t_{iss}). Once it has started the simulation is paused and the Engine is informed about it (t_{tss}). Similarly, the NS will start a simulation with the data received and pause the simulation (t_{ins}). Then the *engine* is informed that the simulation has started (t_{tns}).

After all these steps are completed the simulation is ready to begin. The *engine* will process the events in the queue, and it can be either one corresponding to the SS or the NS.

The total time necessary to complete the initialization phase is as follows:

$$T_{init} = t_{idb} + t_{ien} + \max(2 * t_{tss} + t_{iss}, 2 * t_{tns} + t_{ins})$$

6.3.2 Social simulator event

MASON, the SS used in *Hydra*, and usually most SS, works in steps updating the agents' status every step that corresponds to a time period inside the simulation, in our case, a parameter defined by the user during the configuration of the simulation. This is what I have named SS event. The process of this phase is completed during different steps.

When the *engine* extracts a SS event it creates a message with all the information required by the SS to update its status (t_{ssen1}). This information includes the actual time in the simulation as well as some event predefined by the user that might have triggered.

The message created by the *engine* is then sent to the SS (t_{tss}) that uses this data to update the status of its own simulation what usually translates in the update of the position of the agents (t_{uss}). This changes in the simulation are then stored in a message that is sent back to the *engine* (t_{tss}).

The message created by the SS informs the *engine* that the SS has finished updating its status but some of the changes in the state of the simulation can be necessary in the NS. The *engine* creates a message with the information relevant to the NS (t_{ssen2}) and then sends it to the NS (t_{tns}).

Then the NS uses the data in the message to update the state of its own simulation (t_{uns}). This data can for example contain the new position of the agents that have moved since the previous event in the simulation. The NS then sends a message to the *engine* to inform it that it has finished updated but no more information is included as the NS has not executed new events in its simulation (t_{tns}), it just updated the information of some elements.

Finally when the *engine* receives the update from the NS the execution of the SS event is finished and it can proceed with the next event in the queue.

The total time necessary to complete the SS event phase is as follows:

$$T_{sse} = t_{ssen1} + 2 * t_{tss} + t_{uss} + t_{ssen2} + 2 * t_{tns} + t_{uns}$$

6.3.3 Network simulator event

Differently to how the SS works, the NS events are asynchronous as explained in chapter 5. When the next event in the queue corresponds to a NS event there are different steps necessary to complete until the event is processed.

First the *engine* generates a message with the information necessary to the NS (t_{nsen1}). This information includes the actual time in the simulation and the time when this event ends (this corresponds with the time of the next SS event). This message is then sent to the NS (t_{tns}).

Then the NS extracts the data in the message and proceeds to advance its own simulation during the time specified in the message (t_{uns}). During this simulation time new events will be generated inside the simulation and all the events will be processed as long as their execution time is less than the current event finishing time. An example of events executed during this event could be a router sending a IP packet to a computer that will generate an event when the packet arrives to the computer that once processed might generate new packages sent over the network. When the NS stops updating the simulation (based on the event time define in the message) it then generates a new message with data from the updates and sends it to the *engine* (t_{tns}).

The *engine* then processes this message and generates a new message with information from the updates of the event relevant to the SS (t_{nsen2}). An example of this information could be a notification sent to a mobile phone, so its owner might react to this event. The message is then sent to the SS (t_{tss}).

Then the SS updates its agents with the information received so that it influences future updates of the simulation (t_{uss}). It then generates a message to the *engine* just to inform that it has finished updating the state (t_{tss}).

Finally, once the *engine* receives the message from the SS the processing of the NS event finishes and it can proceed with the next event in the queue.

The total time necessary to complete the NS event phase is as follows:

$$T_{nse} = t_{nsen1} + 2 * t_{tns} + t_{uns} + t_{nsen2} + 2 * t_{tss} + t_{uss}$$

6.3.4 Finalization of the simulation

There are different conditions that can make the simulation end. The user can stop the simulation manually or he may have defined a time or another condition to end. After the ending condition is triggered different steps are necessary to save all the generated data until the simulation can arrive to an end.

Firstly the *engine* sends a message to the SS (t_{tss}) and the NS (t_{tns}) informing about the termination of the simulation.

When the SS receives the message it will start to end its own version of the simulation saving the data that has been generated in a message and then safely stopping the simulation liberating all the data from memory (t_{ess}). The generated data is then sent to the *engine* (t_{tss}). A process in the SS will still keep running in case a new simulation is later started.

When the NS receives the message from the *engine* it will also start to end its own version of the simulation. Similarly to what the SS has done, the NS will save all the data generated during the simulation in a file and proceed to safely stop the simulation liberating all the data from memory (t_{ens}). The stored data includes information about all the packages sent during the simulation as well as information from the different devices. A process in the NS will still keep running waiting for a new simulation to start. The data is then sent to the *engine* (t_{tns}).

When the *engine* has received the information from both the simulation it will analyze the data (t_{een}) and proceed to store it in the database (t_{edb}) so that the simulation can be analyzed later in time. After saving the data, the simulation is complete.

The total time necessary to complete the ending phase of the simulation is as follows:

$$T_{end} = \max(2 * t_{tss} + t_{ess}, 2 * t_{tns} + t_{ens}) + t_{een} + t_{edb}$$

6.3.5 Time aggregation

An aggregation of the time models for the four different scenarios in the simulation (initialization, social simulation event, network simulation event and conclusion) can be summarized in figure 6.6.

$$T_{init} = t_{idb} + t_{ien} + \max(2 * t_{tss} + t_{iss}, 2 * t_{tns} + t_{ins}) \quad (a)$$

$$T_{sse} = t_{ssen1} + 2 * t_{tss} + t_{uss} + t_{ssen2} + 2 * t_{tns} + t_{uns} \quad (b)$$

$$T_{nse} = t_{nsen1} + 2 * t_{tns} + t_{uns} + t_{nsen2} + 2 * t_{tss} + t_{uss} \quad (c)$$

$$T_{end} = \max(2 * t_{tss} + t_{ess}, 2 * t_{tns} + t_{ens}) + t_{een} + t_{edb} \quad (d)$$

Figure 6.6: Simulation times equations

6.4 Conclusions

In this chapter, I have defined the models related to the architecture of the Hydra simulator presented in chapter 5 that is based on the analysis of social and network simulators performed in chapter 3 and the methodology for the creation of co-simulator proposed in chapter 4.

I have described three different models. The first model described has been the data model where I have defined all the different elements that can be found in an AmI environment both related to the social simulation and the network simulation.

The second model is the sequence model where I have studied in detail all the stages of the simulation defining the communication between the *engine* and the simulators. These stages are the initialization of the simulation, the proper execution of the simulation where depending on the time it could be the social simulator or the network simulator advancing the simulation, and finally the ending of the simulation.

The third and final model described in this chapter is a time analysis of the different stages of the simulation in order to know the limitations of time in the simulation.

This chapter has addressed Goal 1: *To obtain a more complete simulation of AmI environments*

using Hydra than executing a social simulator and a network simulator separately, presenting the models used in the simulator that will be used in chapter 9 in the experimentation to validate the contributions to RQ1, RQ2 and RQ3.

Chapter 7

Integration of a simulator in an AmI environment to predict future events

In the chapters 3 through 6, I have used the AmI simulator *Hydra* as a system to analyze the behavior of an AmI environment with *Hydra* being an outside tool that can help obtaining conclusions of the environment for a particular use case. In this chapter I propose an architecture for AmI environments integrating our AmI Simulator, discussed in previous chapters, in a real time environment where the data for the simulations is obtained from real devices placed in an AmI Environment. This can help in several ways such as having simulations more realistic or simulating devices inside the network but in this chapter I discuss the utility of using *Hydra* to generate simulations of a future state based on the current one and then analyze this state to gather some conclusions based on a decision-making algorithm, try to predict events and react to them [Alcarria et al., 2017].

As stated in chapters 3 and 5, a simulator for AmI environments can really help with the deployment and analysis of this kind of environment gathering data from the devices and the people in the environment to be able to extract some conclusion. But this kind of tool can be really helpful once the environment is deployed if it is connected to the network and gathers data from the devices in real time. In this way *Hydra* can generate a scenario based on the current status of the AmI environment and start a simulation to generate a future state of the

environment and then analyze the possible consequences.

I will start analyzing the viability of this approach based on the previous time analysis and get the boundaries on which this can work, then talk about the process of simulating the current scenario and finally introduce the decision-making algorithm that predicts events based on the results from *Hydra*.

This chapter addresses Goal 3: *To predict future events in an AmI environment using Hydra*, presented in section 1.3 and answers RQ5: *Does the integration of a simulator in an AmI environment help in the prediction of future events?* The experimentation that validates this RQ is introduced in chapter 9.

7.1 Simulation integration scenarios

Analyzing how does *Hydra* and their results integrate inside the network, I propose two different possible scenarios: the first one is where *Hydra* is integrated into the same network as the sensors and actuators while the second scenario is where the simulator is not integrated into the network, it is outside it. These scenarios can be seen in figure 7.1.

In the **first scenario**, seen in figure 7.1(a), *Hydra* is integrated into the same network with the sensors and actuators. This enables *Hydra* direct access to the sensor data but, specially more important as seen later in this chapter, direct access to control the actuators. In order for this to work, this also means that *Hydra* needs a module to work as an interface between the simulator and the actuators so that *Hydra* can particularize the desired order for the specific actuator it wants to activate.

In the **second scenario**, seen in figure 7.1(b), *Hydra* is integrated outside the sensors and actuators network. The sensors and actuators communicate directly with a central device and it is this device that then communicates with *Hydra*. In this case the data from the sensors might not arrive as soon as it is generated in the sensors as the central device can process the information or even group it and send it when enough data has been gathered. In addition to

Figure 7.1: Simulator integration scenarios

this problem, the communication with the actuators might also be more complex or even non-existent what could become a problem as seen later because the main idea of the integration is to provide a feedback to the network. The limitations in this scenario can be due to security reasons where all the information of the sensors can't be exposed and the actuators can only be controlled by the central device.

7.2 Data flow model for the integration of *Hydra* into the AmI environment

Until now *Hydra* simulator has worked as an offline tool and as such it has used random generated data or data based on stored data batches from real devices to start the simulation and, based on this, simulated the evolution of the environment over time. But when *Hydra* is integrated into the environment it needs some general data from the environment to begin with

similarly to how a scenario in the simulation is preconfigured. After this configuration, *Hydra* is ready to start gathering data from each of the sensors as it is sent through the network that, as expected, will be asynchronous as not all the sensors in the environment update their status at the same time. Once *Hydra* has stored data from the devices it can start running the simulations to get a prediction of the environment in the near future. Finally this prediction is analyzed and some conclusions are obtained that could result in the influence of the current status of the environment using the actuators that exist within the network in the AmI environment. All the steps in this process are explained in detail in this section and a data flow diagram explaining the process can be seen in figure 7.2.

Figure 7.2: Simulation data flow diagram

The first step is the **connection of *Hydra*** with the AmI environment. *Hydra* needs to be connected to the network where the sensors and actuators are connected and receive all the data this devices generate. This data could arrive directly from the source so that no delay is present as can be seen in figure 7.1(a) but it might not be possible due to the network structure or for security reasons, so the most typical architecture consists of *Hydra* connected to a central device that receives the data from the sensors and actuators as can be seen in figure 7.1(b).

This device can send the data from the sensors to *Hydra* as received or it could provide a rest or similar interface for the data access of external entities to the database . In addition to the data from the sensors, *Hydra* needs access to the actuators so that it is able to give actions to them in order to prevent some future event obtained from the analysis of the future prediction of the current state of the scenario as seen later in this chapter. For this to work, if *Hydra* is connected directly to the sensors and actuators network it needs to have a bidirectional channel, as seen in figure 7.1(a), otherwise if the communication needs to be done through a central device, as seen in figure 7.1(b), there needs to be a way to send actions to the device that are then redirected to the actuators in the network. In case this communication with the actuators is not possible some other way of feedback should be established such as sending an email to the administrator of the network with information of the problem. More on this topic is commented on section 7.3.

Similarly to how a scenario is preconfigured in *Hydra* as seen in chapter 5, the simulation needs to be configured for its use within the environment. The required **configuration of the scenario** includes the layout of the environment, the position of the different devices as well as their characteristics, the periodicity of the simulations executed and the desired simulation time or the ending condition of the simulation. Additionally some information is necessary regarding the people inside the environment. In a normal simulation the scenario is configured with the number of people inside the environment and from there the simulation starts and calculates all the future states. But unless some of the sensors in the environment can count the people inside an area, this information needs to be added manually or an estimation needs to be done based on other sensors that can get a more general quantity of the people such as Passive Infrared Sensor (PIR) Sensors or infrared beams. The problem with this last approach is that a reasonable amount of data is necessary for the calculation to be precise, so the first simulation that can be done should be delayed for a while until the data is gathered.

Once *Hydra* is integrated within the network and configured with data from the environment the first simulation can't start immediately as it would have no data from the sensors in the environment, the next step includes the **calculation of the simulation start** based on the data from the sensors and the previous configuration of the scenario as can be seen in figure 7.2.

Hydra needs to wait some time until enough data has been recollected from the sensors and if possible until at least one data has been obtained from each of the sensors in the network. As *Hydra* has been configured with the number and kind of devices in the network it could potentially wait to receive data for all of them to start the first simulation, but this could cause a potential problem where one of the devices updates its information in a long period of time compared to other devices in the network or even worse if one of the devices has disconnected or otherwise is no longer available. Taking all this into account, the simulation should start once there is data from enough devices to avoid being waiting for a long time and start analyzing the near future of the environment even if the prediction is not very precise due to the lack of information from some devices.

In addition to the problem of getting data from all the sensors, starting a simulation in a particular moment can create a problem where *Hydra* is using recent data from some devices but old data from others due to the difference in the periodicity of the updates from the sensors. There are three different options to approach this **difference in time data** problem:

- Use the most current data from each device: as explained before, this could create an inconsistency in the simulation status of the different elements as both, a data that has been received several seconds ago and a data that was last updated hours ago could be used.
- Use a projection of the data: instead of using the most recent data from each device, one new data could be generated for the current time based on the data history of each device. Depending on the quantity of data gathered and the variability of each sensor this data could be more or less precise to the actual value.
- Use older data: from the the devices that have a shorter periodicity older data could be used so that the variance of time between all the data is minimal. This could add a problem where a device hasn't updated its value since the last simulation or worse if the device has disconnected.

In section 7.4 I will detail how the data is generated and how it is decided to start or delay a

simulation.

In addition to the time for the simulation to start, another important parameter for the simulation is its **ending condition**. It will generally be an elapsed simulation time but similarly to a normal simulation it could be the simulation state reaching a particular predefined condition. As previously commented the objective of the integration is obtaining a near future state of the environment based on the current one and then the result is analyzed and *Hydra* can act accordingly. All this process should take as little time as possible, so in any case the simulation time shouldn't be very long for the results obtained to be relevant in the actual context. For example if I want to generate the prediction of the environment in the next minute, the simulation won't be useful if it takes more than one minute to process. All this time analysis is done in more detail in section 7.5.

Once the simulation initial parameters are obtained, including the ending condition and the data from the sensors, the **simulation** can start in the very same process as described in chapter 6 obtaining the future state from the social simulator and the network simulator.

When the simulation ends, then the **future prediction scenario** is processed by the decision-making algorithm as seen in the section 7.3. This algorithm will analyze the scenario to gather some conclusions and if some critical event is detected, then act accordingly. In addition to this the calculation of the next simulation starts using this state as base for the calculation.

The simulation process repeats itself over time periodically so that the environment is continually monitored and the appropriate actions are taken if necessary. Depending on the use case desired for *Hydra*, the complexity of the scenario, the time it takes for the sensors to update their status and the results of the previous simulation, the next simulation could start in any desired moment. In a critical scenario the next simulation could even start as soon as the previous one has finished without waiting for the analysis. At least some of the sensors in the environment should have updated their status, so that the next simulation gathers a possibly interesting conclusion, but ideally all of them should have updated their state. Anyway, as discussed previously in this section, it might not be a good idea to wait for all of the sensors to have sent new data because there might have been some problems and some of them might

have stop worked, but it is a good idea to wait for the majority of the sensors to have updated the information as long a reasonable time has passed from the previous simulation. In addition to this, as *Hydra* recollects all the data from the devices in the AmI environment, once it has enough data it can extract some conclusions from the data such as the periodicity of those devices. So, for example, if a device has sent data in minute one, in minute 2 and minute 3, and now it is minute 3.5, *Hydra* could expect this device to send the next data in half a minute. This information can be helpful for calculating when to start the next simulation so that *Hydra* can choose the best time based on that information.

7.3 Decision-making and predictive solutions

Once the simulation has generated a state of the near future of the AmI environment based on its current status, there is a process to analyze this new state and try to gather some conclusions of what is about to happen. Based on certain predefined patterns and the history of the simulation of that AmI environment this module can predict events that might happen. Depending on the desired use case of *Hydra* these events will be different and can include things like detecting that a fire is going to happen, or there are going to be more people than allowed inside a place, that the network is going to saturate or that enough people are going to see the advertising panel. Other events not directly related to the use case but interesting anyway could also be detected, such as a device running out of battery.

How the module is trained to detect the patterns is out of scope of this chapter as there are already lots of algorithms proposing general solutions but I will introduce some of them that could be useful. *Hydra* gathers all the data from the sensors so based on the historical data of a device, several patterns could be studied specially if for example a device has stopped sending data it could have disconnected from the network or probably run out of battery, so maybe studying the behavior of the last times it sent data could gather some conclusions such as missing some of the periods sending data or sending corrupted data.

The module can be trained to detect these patterns and if any are detected in a simulation,

a new process will begin to act accordingly as defined by certain rules. As explained on the previous section, depending on how *Hydra* is connected to the device network there will be different actions that can be taken to give feedback to the network about the results obtained. If *Hydra* has direct access to the actuators in the network, the simulator could activate any relevant one to act according to the results of the simulation. So for example if a future fire is detected an evacuation should take place and trigger the fire hooter in the building could work as a prevention mechanism. If our use case is the targeted advertising *Hydra* could change the advertising panel accordingly.

Until now I have only considered the option of analyzing the final scene simulated but it could be possible to process every state *Hydra* generates before getting to the final one, once the ending simulation time occurs. The problem with this approach is that it might not be feasible as the total simulation time would increase due to the addition of the processing after each cycle in the simulation is generated. Another option is the parallelization of both processes so that the simulation doesn't stop after each cycle is complete. This would only make sense if both processes take place in different computers or in the same one but in different cores. Otherwise the execution of one would slow down the other.

7.4 Data processing

As seen in the data flow diagram in figure 7.2, one of the core modules of this simulator set up is the Data processing. I have previously shown some details in chapter 5 where I specify the *data manager module* in the normal simulator environment pointing out that there were more capabilities when *Hydra* was integrated into a real AmI environment. In this chapter the *data manager module* has the important task of processing and storing all the data received from the sensors as well as adapting the actions to communicate with the actuators in the environment.

Firstly the *data manager module* needs to **store the data from the sensors** keeping a historic record of the different values over time. These data will be used to feed *Hydra* with real data from the sensors but not only the most recent one is going to be used, the older ones will be

used to predict the actual value of the data and the evolution of it over time.

The main component that needs the data from the sensors is the process that calculates the initial start time and data for the simulation of the AmI environment. As seen in section 7.2 this process needs the data to start the simulation but the time to start the simulation is based on the availability of this data and their reliability, as well as results from the analysis of the future scenario prediction. The simulation start process asks the *data gathering module* periodically information about the status of the different sensors. The *data gathering module* responds with the last data obtained from the network as well as a reliability parameter depending on the last time this data was updated and the variability of the data based on previous information. The reliability parameter is a key one for the simulation start as depending on how critical a sensor is and the parameter the simulation can be started or postponed as necessary.

In addition to obtaining the last data from a sensor, this module can also return a **processed value** for each sensor. This value will be based on an estimation of previous recent data as well as the evolution of the data in the sensor in similar days and time. So for example if the temperature was detected to lower in the previous day during night and it is at night one can also expect a decrease in the temperature. In addition the data can be estimated from other similar sensors or sources of data that in the surrounding environment. So for example, an increase in temperature can be estimated if more people is detected getting inside the room or if the temperature outside the building is rising. Again all this predictions result in a reliability parameter based on the data available and the decisions taken.

7.5 Assessment of the simulation model

The solution proposed in this chapter includes several assumptions under which the solution can be valid. In this section I discuss what are the requirements for *Hydra* to be useful in a real AmI environment and in the next section I use some example scenarios with *Hydra* to get some real times.

In section 6.3 I have done a time analysis of our simulator in order to get the required time for

each of the different phases in the simulation: initialization, social simulation event, network simulation event and finishing the simulation. The summary of the different times can be seen in figure 7.3.

$$T_{init} = t_{idb} + t_{ien} + \max(2 * t_{tss} + t_{iss}, 2 * t_{tns} + t_{ins}) \quad (a)$$

$$T_{sse} = t_{ssen1} + 2 * t_{tss} + t_{uss} + t_{ssen2} + 2 * t_{tns} + t_{uns} \quad (b)$$

$$T_{nse} = t_{nсен1} + 2 * t_{tns} + t_{uns} + t_{nсен2} + 2 * t_{tss} + t_{uss} \quad (c)$$

$$T_{end} = \max(2 * t_{tss} + t_{ess}, 2 * t_{tns} + t_{ens}) + t_{een} + t_{edb} \quad (d)$$

Figure 7.3: Simulation times equations

In this chapter now the analysis becomes very relevant as for a simulator to work in a real time environment, first the simulation has to finish in a reasonable time, considering a reasonable time at least before the desired simulated time becomes a real time, but usually it will be necessary to finish sooner or there won't be enough time to react to the conclusions extracted from the simulation. How much sooner is necessary to finish the simulation will depend on the desired use case of the integration of *Hydra* into the AmI environment. For example, if one wants to prevent a fire or evacuate the building in a short time, the simulation will need to finish very quickly, usually in no more than a few seconds, but if I want to display targeted advertisement depending on how many people there is going to be in a room, there shouldn't be a problem if the simulation takes a while longer. In the next section I particularize these times for some scenario examples with our simulator *Hydra*.

In order to get a longer or shorter simulation the parameter **Tstep** presented in section 6.2 can be varied. This parameter is the simulation time that spans between one cycle of the simulation and the next, considering a cycle the time between one update of the elements in the simulation and the next (both in the network simulator and the social simulator). The smaller this parameter is, the more real the simulation is as the movement of the elements in the environment can be calculated more precisely and the elements react sooner to other elements movement or events, so for example a person can react sooner to its mobile phone getting in range of a Wi-Fi network. The problem with a shorter Tstep is that the simulation will also be more consuming in both computing resources and time, as more cycles are needed to

complete the simulation. On the other hand, the longer Tstep parameter is, the less consuming the simulation is in both resources and time but there will also be more errors in the simulation as the calculation of the movement of the elements in the environment will be less precise and it will take more time for the elements to react to other elements movement or events. The time it takes for a simulation cycle to complete varies on the complexity of the scenario and therefore on the complexity of the simulation. A scenario with lots of devices and people will take a long time to simulate so a larger Tstep will be needed to achieve a reasonable simulation time, while a simple scenario with just a couple of devices and people in it will take much less time and Tstep can be set to a smaller value to achieve a simulation with less errors. In the equations in figure 7.4 you can find the cycle time, the total real time a simulation takes, the simulation time and the time required for the simulation to be valid.

$$T_{cycle} = T_{sse} + T_{nse} \quad (a)$$

$$T_{total} = T_{init} + N * T_{cycle} + T_{end} \quad (b)$$

$$T_{sim} = N * T_{step} \quad (c)$$

$$T_{total} < T_{sim} \quad (d)$$

Figure 7.4: Simulation times requirements

Going back to the viability of using *Hydra* for future events prediction, this will depend a lot on the complexity of the scenario, being reasonable on a small scale scenario, but probably unfeasible in a large one as it would require a very large time to execute the simulation. As previously commented it will also depend in the desired use case of the predicted events as some scenarios such as an evacuation will require a shorter reaction time while in some others, such as the massification of a place to check Wi-Fi coverage, a larger time might not be a problem or at least not too critical.

In chapter 9 I will do this analysis with 3 different scenarios using *Hydra* to get an idea of the limits in this approach in different contexts.

7.6 Conclusions

In this chapter I have analyzed the integration of *Hydra*, our AmI simulator, into a real AmI environment in order to feed *Hydra* with data from real sensors and be able to predict future events that may happen in the environment and react using the actuators.

I have presented the different possible scenarios in which *Hydra* can be integrated, I have detailed the different modules that enable the integration and I have finally analyzed the viability of the integration depending on the characteristics and complexity of the scenario.

In this chapter I have addressed Goal 3: *To predict future events in an AmI environment using Hydra*, presented in section 1.3 and answered RQ5: *Does the integration of a simulator in an AmI environment help in the prediction of future events?* The experimentation that validates the contributions to this RQ is introduced in chapter 9 where I will measure the simulation of a scenario varying certain parameters to assess the analysis performed in section 7.5.

Chapter 8

Use of prosumerization for the simulation of Ambient Intelligence environments

In this chapter I propose an architecture that interconnects the simulator Hydra presented in previous chapters with a semantic platform and add a prosumer framework to enable end users an easy interaction with the simulator. This framework enables the creation of services that gather specific data from the semantic platform and feed the simulator with real data.

In this chapter I propose a prosumer framework for easy service creation by end users with no experience creating and deploying services over a semantic AmI platform taking advantage of knowledge from several AmI deployments domains [Martín et al., 2017]. This prosumer platform will enable the creation of services where any person with no knowledge about ontologies, models, service composition or AmI architecture; can create visually functional services for running them over a semantic AmI platform regardless of the testbeds that are bellow, and allowing services portability across platforms. This prosumer framework offers services creation and deployment, and also the design of service templates allowing services portability from one environment to another in just a click, optimizing the performances of services running, as well as the data consumed itself by each service. This platform will allow the consumption and use

of data from the a semantic AmI platform in a much more accessible and flexible way for end users; making it a “deployment-Agnostic query” system. The services then enable the user the use of the data to run his own simulations.

This chapter addresses Goal 4: *To improve the user experience with Hydra applying prosumerization techniques* presented in section 1.3 and answers RQ3: *What is the perception in the use of the co-simulator?*, RQ6: *What is the quality of the services developed by prosumers?* and RQ7: *Who are more efficient: prosumers or expert developers?* The experimentation that validates these RQs is introduced in chapter 9 where I will analyze the boundaries of integrating Hydra with an AmI environment.

8.1 Semantic platform

I have used a platform with the main aim of sharing and reusing data form diverse semantic platform deployments in a flexible way. The semantic platform integrates several IoT platforms, deployments (also called testbeds), it opens up new chances in the research, development and deployment of research works that take advantage of data and capabilities. The semantic platform aggregates four core deployments which are a large-scale Smart City deployment, a smart environment based on an indoor sensors nodes deployment, a large-scale crowd-sensing testbed and an indoor and outdoor building smart environment deployment. The semantic platform proposes an ontology that aims to achieve semantic interoperability among heterogeneous deployments, allowing to seamlessly support the federation of testbeds using semantic-based technologies [Gyrard et al., 2015].

I am going to focus on a very small part of the ontology proposed by the semantic platform I have been using; considering the following elements: deployments, devices, and observations. The data model used in this research; where a deployment is an aggregation of devices, and a device has many observations.

8.2 Prosumer framework

I define a prosumer framework where a non-expert in the semantic platform architecture nor expert in the AmI platform architecture can easily develop services over the the semantic platform also taking advantage about the information and data offered by the the semantic platform ontology. User participation is a new trend today. Participants in this environment evolve from content consumers, who use the the semantic platform to find for important information for their work and research and also to produce new knowledge and create services, making them available to other users. Figure 8.1 shows in red the elements developed in this research work, while blue elements show the assets where our proposal will be supported from the semantic platform architecture. Our research benefits from semantic technologies not only for the provision of interoperability among federated testbeds but also due to the potential of these technologies in the derivation of further higher-level knowledge. The prosumer framework offers a visual web front-end programmed using Hypertext Markup Language (HTML)⁵, Cascading Style Sheets (CSS)³ and JavaScript; in order to facilitate end users the creation and deployment of their own services in a simple way.

The prosumer framework enables the user the selection of different elements to obtain the desired data for the simulation defining rules. A rule is composed of an event (the device

from where the data is obtained), a possible condition (a filter to apply to the data), and an action. A service is composed of one or more of these rules. The information is obtained in real time through the semantic platform query system, that is, the set of deployments, devices and observations that the prosumer framework offers to its users are obtained on running time by asking SPARQL Protocol and RDF Query Language (SPARQL) questions to the queries systems of the semantic platform.

In order to define the event there are 2 steps the user needs to define:

1. First the user has to select from which of the 4 testbeds they want to gather the data (these are a smart city, a smart environment, a large-scale crowd-sensing testbed and an indoor and outdoor smart environment). As currently there are 4 testbeds these are the one included in the prosumer framework but if more appear in the future they could be easily integrated in the platform. Once a testbed is selected, the different devices that belong to that testbed appear.
2. Then the user has to select the device from where he wants to gather data. Examples of these devices include sensors such as ambient temperature, illumination or presence. Each device is identified by an ID and several data is associated to it such as the kind of sensor it is, the location or the type of data values it offers and this data is presented to the user once the device is selected.

Then the user defines a condition. This lets the user filter the data generated choosing a date range or a data range generating a subset of the data he wants to work with, or even accessing the data in real-time, generating a stream of data for the simulator to work with.

After defining the event and the condition, the user needs to define the action or actions he wants to take with that data, if any, and the desired order of these actions. These are four different currently available options, but the set of actions will be increased as the prosumer framework requirements increase:

- **Soften:** Specify a boundary (defined by at least one between a minimum value and

Figure 8.2: Prosumer framework screenshot

a maximum value) and all the data that falls out of it will be rounded up or down accordingly.

- **Operation:** Apply a multiplication, division, sum or subtraction to the data.
- **Frequency:** Change the frequency of the data so that new data is generated if a higher frequency is desired or some is discarded if a lesser one is selected.
- **Value Deviation:** Change the value between a percentage or a fixed range.

An image of the prosumer environment and the process where the user defines the event, condition and action can be seen in figure 8.2.

Each of the rules are transformed into a SPARQL query with a specific execution frequency. This query (and its execution frequency) is sent to the “semantic platform service execution

support system”, which returns the results of the SPARQL query with the indicated frequency to the prosumer framework “service provision environment”.

The service provision environment consists of a visual web application dashboard where a prosumer can start, pause, stop, resume, modify and delete rules running over the “semantic platform service execution support system”. I also call it the prosumer framework engine, since it is a server that is responsible for the execution of each of the rules created and to attend to the requests with the results of SPARQL queries from “semantic platform service execution support system”. In case the rules are real time, the server keeps running the service until it is stopped or paused and in case the rules are not in real time a file with the data processed is generated and can be downloaded. All the rules in a service need to be of the same type, whether real-time or not.

8.3 Conclusions

In this chapter I have integrated the simulator Hydra with a semantic platform using a prosumer framework to access the data in the platform. This has enabled that the data from the simulator is obtained from the platform rather than generated in the simulator.

This chapter has addressed Goal 4: *To improve the user experience with Hydra applying prosumerization techniques* presented in section 1.3 and answered RQ3: *What is the perception in the use of the co-simulator?*, RQ6: *What is the quality of the services developed by prosumers?* and RQ7: *Who are more efficient: prosumers or expert developers?* The experimentation that validates the contributions to this RQ is introduced in chapter 9. In the next chapter I will propose a methodology for the creation of co-simulators based on the analysis done in this chapter.

Chapter 9

Validation

Having presented all the contributions done for this PhD Thesis, all this work is meaningless if it is not properly validated. In this chapter I present the experimentation done to validate the different goals and RQs proposed in section 1.3. The goals are the following:

- **Goal 1:** *To obtain a more complete simulation of AmI environments using Hydra than executing a social simulator and a network simulator separately*
- **Goal 2:** *To propose a methodology that will help in the creation of co-simulators for AmI environments*
- **Goal 3:** *To predict future events in an AmI environment using Hydra*
- **Goal 4:** *To improve the user experience with Hydra applying prosumerization techniques*

The following sections present the validation carried out for each of the goals and their associated RQs. Section 9.1 addresses goal 1 and RQs 1, 2 and 3 described in chapter 3. Section 9.2 addresses goal 2 and RQ 4 described in chapter 4. Section 9.3 addresses goal 3 and related RQ 5 described in chapter 7. Finally section 9.4 addresses goal 4 and RQs 3, 6 and 7 described in chapter 8.

A diagram of the relations between the goals, RQs and experiments performed can be seen in image 9.1.

Figure 9.1: Proposal of architecture for a CPSS co-simulator for AmI environments

9.1 Proposal of an Ambient Intelligence simulator

In chapter 3 I have presented a proposal for an AmI simulator based on the analysis done previously in section 2.4 about the features found in social simulators and network simulators.

In this section I address the associated validation where I have carried out a case study in order to address goal 1 "To obtain a more complete simulation of AmI environments using *Hydra* than executing a social simulator and a network simulator separately" and the following RQs:

1. RQ1: Does the network and social co-simulation offer more features than the simple execution of each one separately?
2. RQ2: Does the network and social co-simulation help in the analysis and drawn of conclusions?
3. RQ3: What is the perception in the use of the co-simulator?

During the experimentation step four experts evaluated the use of *Hydra* and the execution of *MASON* and *ns-3* separately analyzing the use of these simulators in three specific scenarios. In addition, twenty one students from an AmI course filled in a survey about the satisfaction using *Hydra*.

With this approach I analyzed a set of events triggered as a consequence of users' actions (both experts and students) with the *Hydra* co-simulator in a real-life setting in order to identify the

factors that influence the simulations. The researchers that carried out the case study had little control of the participants using *Hydra* because they had only trained the participants in the use of *MASON* and *ns-3* and solved their problems during its use. This approach is appropriate to replicate the experiment in similar contexts.

9.1.1 Context

The case study was designed, planned, guided, monitored and evaluated by the experts, who have more than five years of experience in network and social simulation, cloud computing, etc.

To answer **RQ1**, experts qualitatively analyzed the characteristics of *Hydra*. In this study they modeled three AmI scenarios in *Hydra*:

- **Scenario 1:** The evacuation of a building. This scenario is based on an AmI evacuation system based on dynamic routes [Morales et al., 2014].
- **Scenario 2:** The Wi-Fi coverage of a building and the movement of people around the building searching for connectivity with their mobile phones.
- **Scenario 3:** In a university school, the exit from class and the saturation of the corridors in the building at that time of the day.

To answer **RQ2**, the experts analyzed the number of conclusions drawn from the execution of simulations of the three scenarios described above. I define a **conclusion** as something relevant extracted from the simulation in the expert's opinion; for example in the scenario 2 some conclusions could be the high concentration of people in some rooms or another conclusion could be the lack of Wi-Fi coverage in some areas. Two types of simulations were conducted: the first one executing the simulators separately (*MASON* and *ns-3*) and the second one using *Hydra*.

RQ3 was answered with the participation of twenty-one people (hereafter, learners) at "Universidad Politécnica de Madrid" (UPM) that carried out the case study proposed in this research

work. They are graduates, mostly in telecommunications engineering, and Master students. They were collaborating with the "Grupo de Investigación de Servicios Avanzados de Internet" (GISAI) group at UPM in the area of a subject of the Master related with AmI in order to test *Hydra*. These learners had to model some AmI scenarios using the approach proposed in this paper. They were surveyed by the experts in order to evaluate the satisfaction with the use of *Hydra*.

9.1.2 Plan

The case study was split in two experimentation tracks: the first track involved the experts and the second one the learners (see figure 9.2). Each track had very similar phases but with different objectives. These phases were:

Training phase: The first phase consisted in training the learners in the principles of social and network simulations and its benefits; they also were trained in the use of the *Hydra* tool and its capabilities. The training sessions consisted of 2 hours of theory at the beginning of the experimental validation with personal interviews to solve specific problems found.

Simulation execution: This phase was carried out by both groups, experts and learners but with different objectives for each one. Both groups modeled three different AmI scenarios. Experts used *Hydra* and then MASON with ns-3 to model the three scenarios, while learners only used *Hydra*. The difference between the groups consists on the experts discussing the differences between using *Hydra* against MASON and ns-3; while trainees only study the subjective perception of use and satisfaction of using only *Hydra*.

Evaluation phase: At this phase all the scenarios modeled and their conclusions were assessed and evaluated by the experts as well as the surveys performed by the learners.

Figure 9.2: Case study plan with inputs and outputs

9.1.3 Data gathering

All data was obtained during the evaluation phase. On the track performed by experts an analysis of the qualitative characteristics of the proposed approach in this work was analyzed in order to answer RQ1; checking whether the proposal really meets the requirements presented in the section 2.4. In order to answer RQ2 the experts also analyzed the number of conclusions obtained after executing three simulations (one per each scenario proposed) using both sets of simulators: *MASON* with *ns-3* and then *Hydra*. In this evaluation, if the same conclusion was obtained by analyzing the results of *MASON* and *ns-3*, it only counted as a single conclusion. In order to avoid learning among the experts, scenarios and conclusions; each scenario was simulated by two experts using *Hydra* and two different experts using *MASON* and *ns-3* separately.

As previously defined a **conclusion** is considered as a judgment obtained after the evaluation of the results (or partial results) proposed by the execution of a simulation; and a judgment is considered as a relevant fact for the scenario executed in a simulator.

On the track performed solely by learners, they were surveyed about the use of *Hydra* and its personal subjective satisfaction with its use; they were also asked about their subjective perception about the execution steps of *Hydra* [Sánchez-Picot et al., 2016]. They were surveyed using a five-level Likert scale for both questions.

9.1.4 Results

This section presents and discusses the results obtained in the case study and will be presented according to the research questions proposed. Firstly an analysis is performed by the experts about the qualitative features of *Hydra*. Next the conclusions obtained by the execution of the two simulators (*MASON* and *ns-3*) separately are shown and the approach proposed in this research in order to test whether the proposal discovers more relevant conclusions for the execution of AmI scenarios. And finally, a study of users' satisfaction using *Hydra* is presented showing the overall assessment of the tool and an evaluation of the steps proposed by the approach for simulating AmI scenarios.

Qualitative analysis of the characteristics of Hydra

This section examines whether the desirable characteristics of AmI scenarios simulator have been satisfied by *Hydra* as can be seen in Table 9.1 .

The experts agree that all the models are currently included in the simulator. Some of them require an extension, such as the mobility model that currently has very few different possibilities or the group behavior that also requires an improvement.

The execution features are nearly complete with the exception of the scene design tool as has already been commented in chapter 3 and is still being developed, The batch execution is also a work in progress as the efforts are currently in the visual simulation and the integration into

Table 9.1: AmI simulator features fulfilled

Feature	AmI Simulator
Environment modeling	✓
Mobility modeling	✓
Individual behavior modeling	✓
Group behavior modeling	✓
Human interaction modeling	✓
Physics modeling	✓
Network protocol modeling	✓
Network device modeling	✓
Preconfigured scenes	✓
Parameters configuration in scenes	✓
Scene design tool	×
Real-time execution	✓
Batch execution	×
Real-time modification	✓
Integration into Real-time systems	×
Discrete-event simulator	✓
Real-time analysis	✓
Post execution analysis	✓
Visual analysis	✓
Text logs	✓
Batch data analysis	×

a real time environment that still requires some adjustment.

The analysis features are also fulfilled, with the exception of the batch data analysis. Despite this, some of the analysis tools still require much more development, specially the post execution analysis that nowadays lacks many features.

Number of conclusions drawn using Hydra

This section presents an analysis about the number of conclusions obtained after the simulation of the three scenarios using MASON and ns-3 on the one hand and *Hydra* on the other hand. An expert simulated each scenario using Hydra or MASON and ns-3, but never both; in order to avoid learning. As explained in 9.1.3, duplicates found both while simulating in MASON and ns-3 are merged and we also trust the expert's judgment for the relevance of the conclusions.

Table 9.2 presents the number of conclusions drawn by each expert and scenario. It can be seen that the number of conclusions is slightly higher when Hydra is used to model and simulate

Table 9.2: Conclusions extracted by experts.

# Expert	Scenario 1		Scenario 2		Scenario 3	
	Hydra	Mason&NS3	Hydra	Mason&NS3	Hydra	Mason&NS3
Expert 1	×	3	5	×	×	5
Expert 2	×	4	×	4	6	×
Expert 3	6	×	6	×	7	×
Expert 4	6	×	×	4	×	6

AmI scenarios. The total number of conclusions drawn using Hydra was thirty six, while using MASON and ns-3 twenty six conclusions were obtained (see Figure 9.3).

In this case the difference is only ten conclusions, which may seem few; but the important thing is that using an AmI simulator as *Hydra* there are some conclusions that can be drawn that are impossible to extract otherwise.

Figure 9.3: Total of conclusions drawn

These results suggest that the use of a co-simulator as *Hydra* can provide more relevant conclusions for the simulation of AmI environments than simply using a social simulator and a network simulator.

Analysis of the users perception with an AmI environment simulator

The twenty one learners who participated in the case study evaluated their satisfaction with the *Hydra* simulator, and were asked about their subjective perception of the utility and the need for an AmI environment simulator (see figure 9.4).

Figure 9.4: Overall evaluation about the utility of a co-simulator

The mean of evaluations is 2.94 and the standard deviation is 1.07; this means that assessments have been generally very high and learners have evaluated it as very useful tool.

Anyway there are several low scores for the utility, meaning that learners have evaluated the tool as unhelpful, this may be due to the difficulty of understanding the need for a simulator of AmI environments; since, in an AmI environment, it is very difficult to differentiate between the social and the network sides and the advantages of a co-simulator.

Learners were also surveyed about the four steps proposed by *Hydra* to perform a simulation [Sánchez-Picot et al., 2016] as explained in 9.1.1. This survey was conducted in order to study what steps can be improved in *Hydra*. Table 9.3 and figure 9.5 presents the results obtained.

Table 9.3: RQ3 evaluation summary.

Step	Evaluation	
	Mean	SD
Step 1	2.43	0.75
Step 2	3.62	0.92
Step 3	3.76	0.7
Step 4	1.95	0.59
Total:	2.94	1.07

Figure 9.5: Evaluation of the steps proposed by Hydra to perform a simulation.

As can be seen, steps 2 and 3 were the most highly valued by learners while steps 1 and 4 were the worst rated, especially the fourth step. This is because in step 1 there are still very few preprogrammed scenes and there is currently not a way to create new ones. Step 2 seems to have enough configuration options and similarly step 3 is solid enough with the ability to inspect elements and change the values. The analysis tool in step 4 shows very few options, so much more is expected for a proper analysis.

9.2 A Methodology for the Design of Application-Specific Cyber-Physical Social Sensing Co-Simulators

In this Section, a practical validation for the methodology proposed in chapter 4 is provided addressing Goal 2: *To propose a methodology that will help in the creation of co-simulators for AmI environments* and RQ 4: *Does a methodology help in the creation of AmI co-simulators?*.

9.2.1 Experimental validation: Co-Simulator Development and Experiment Description

In this subsection an experimental validation for the co-simulator is presented. A particular CPSS co-simulator is designed and implemented using the proposed methodology, and, later, some experiments are designed in order to evaluate its performance and the invested time in its development comparing the data with those obtained from other solutions. The rightness of the proposed solution will be deduced from the results obtained.

Co-Simulator Implementation

For this use case of the proposed methodology, a CPSS co-simulator focused on validating crowd management systems in large facilities for emergencies such as panic control or enhancing attention is designed. There are three main subsystems: the **physical world** such as walls, stairs or doors, the **cyber world** including sensors and actuators and the **social world** that includes the people that moves around the environment, their behavior and relations. There are transversal components such as time constraints that condition the whole system's operation and should be taken into account in all subsystems and domain particular simulations.

STEP #0. Final Users' Requirements and Characteristics Capture

First we obtain the requirements and characteristics of phase 0, particularly the requirements of the four properties described in section 4.2:

- REQ#1. Flexibility: There are no specific requirements about flexibility because the proposed simulation tool is focused in only one application, that is, crowd management.
- REQ#2. Modularity: New types of devices should be included in the future as new technologies appear.
- REQ#3. Scalability: The co-simulator is going to focus in large facilities scenarios meaning several thousands of people. We define a maximum of fifty thousand agents but the most common case will be much lower.

- REQ#4. Accuracy: There is no need of a high level precision in the designed tool because social models incur a limited accuracy as human behavior prediction is complicated.

Additionally, simulation tools specific for crowd management have some particular characteristics:

- CHAR#1. The simulation should not require technological knowledge because they are executed by social experts.
- CHAR#2. The crowd management global state, such as if the panic is growing between the people, is more important than specific values at a certain time.
- CHAR#3. In crowd management systems the most relevant subsystem is the social world with the cyber and physical worlds being secondary.
- CHAR#4. It can be hard to have a precise model of the building as they are complex structures. Additionally lots of agents are in the simulation they share the same behavior, so the required processing capabilities to execute the simulations are limited.
- CHAR#5. The number of people inside a building is limited first by the maximum allowed capacity but even at overcapacity there is a limit so, similarly the number of agents that can coexist in a scenario is limited.
- CHAR#6. The results need to be represented with temporal and statistical graphics and with animations that show the scenario evolution.

Additionally to the commented characteristics, there are certain circumstances that need to be taken into account to implement a co-simulator related to the final users and development team:

- CHAR#7. In the team of future developers there is no expert in simulations, so a complicated and specific implementation can't be done.

- CHAR#8. Because the scenarios are focused on large facilities there is no need for users to be able to design their own scenarios.
- CHAR#9. For this particular case the used domain-specific simulators have been: MATLAB/Simulink as physical processes simulator, ns-3 as network simulator and MASON as social simulator. This selection has been done because the extended use of these simulators in research, due to their open architecture, because MASON and ns-3 are open source and also because their efficient performance.

With the base established in this step, the rest of the steps in the methodology can be applied for the creation of the co-simulator.

STEP #1. Selection of the co-simulation paradigm

In this step the appropriate co-simulation paradigm will be selected taking into account all the possibilities seen in section 4.3.

First the following considerations need to be done related to the characteristics defined in step zero:

- As commented there are no expert programmers on simulation (see CHAR#7), so integrated co-simulation was discarded (see section 4.3).
- All the considered simulators have an open architecture (see CHAR#9), so the possibility of federated co-simulations exists. MATLAB is not an open source tool so choreographed co-simulation is not an option.
- The results presentation and graphic interface provided with the chosen tools don't meet the requirements of a crowd management system, and as seen in CHAR#6 any environment that displays animations can be used.
- Users are not programmers (see CHAR#1), so the proposed co-simulator needs to be implemented using the orchestrated by a third party engine federated co-simulation paradigm.

The architecture of the proposed co-simulator based on the characteristics commented previously can be seen in figure 9.6. As an additional component the database that stores all the logs is included. ns-3 simulator has a modular architecture so REQ#2 is satisfied and any new device that may appear can be easily introduced in the co-simulator. Finally, all the domain-specific independent simulators meet the scalability requirement REQ#3.

Figure 9.6: CPSS Simulator Architecture

STEP #2. Particularization of the general simulation model and simulation life-cycle

For the co-simulator being built the simulation lifecycle seen in figure 4.4 is not changed. There is no need to add other tasks as indicated in the characteristics previously and all the states are needed.

Related to the simulation model, the most important subsystem in the co-simulator is the crowd management system in social world as stated in CHAR#3, what means that it should be extended. Additionally the physical world should be also complemented to represent the scenario properly. Taken this into account, the model is extended as can be seen in figure 9.7 the social part and in figure 9.8 the physical part, compared to the original model presented in figure 4.3. The relations of "Personal state", "Social state" and "Physical state" are the new

components in the model.

Figure 9.7: Simulator#1 architecture

Figure 9.8: Simulator#2 architecture

Related to the cyber world, ns-3 simulator needs to simulate all the required device in the CPSS environment and specifically control, communication and processing devices as seen in figure 4.3. Several works have proved that ns-3 can be used in such a way [Alcarria et al., 2014], simulating anything from huge computing systems to small wireless sensor networks and cellular communications.

STEP #3. Selection of the appropriate coordination mechanism

In this step the coordination mechanism is selected. There are three different simulation schemes in the co-simulator:

- The physical simulator MATLAB has a continuous simulation sharing the results as soon as the process finishes.
- The social simulator MASON also has a continuous simulation sharing the results each Tstep.

- The network simulator ns-3 has an event-oriented simulation.

At the implementation stage these three different schemes can be reduced to only two different ones: in the physical processes simulators it is typical to get the simulation routing with a program that runs the simulation each time step showing the processed results. So, in this case MATLAB and MASON behave similarly. Examining this situation, implementing a parallel execution solution in the co-simulator is not viable.

The necessary processing capabilities to process the simulations are limited as seen in CHAR#4. Additionally the number of agents in the simulation scenarios is limited as seen in CHAR#5. Considering this, the simulation times should be small so stop-and-wait mechanism is the preferable solution for this co-simulator. Now, the specific design of the proposed coordination mechanism is explained: First the engine gets the global model for the simulated scenario keeping track of the future events that are happening in the three domain-specific simulators. The engine has an event queue where this information is stored and the events are ordered by time and always the first is extracted to be processed. ns-3 calculates the future events in a row when the simulation starts, and sends it to the engine that stores those events in the event queue. MASON and MATLAB update the state of their simulation every period of time, Tstep. The events that happen during this Tstep are also stored in the event queue. The value of Tsteps needs to be defined at implementation stage to meet REQ#4 related to the tool's accuracy. For this co-simulator this value has been set to one second.

After the three simulators have been properly configured, they pause their simulations and only continue them when the engine dictates so. The engine monitors the first event in the queue and sends a message to the simulator that event belongs to so it can continue with the next step or process the next event accordingly. Then the simulators execute their process and once finished the data generated is sent back to the engine. This data can be relevant to other simulators so the engine processes it and informs the other simulators accordingly.

When the simulation finishes all the data and logs generated during the simulation is stored in the data base what achieves REQ#3 related to scalability. The simulation can take very long time, but hold perfectly with several thousands of agents.

As a comment, the proposed solution is one of the many possible ones. The scheme proposed could not be applicable in other situations or scenarios. There exists also the possibility of continuous-time simulators using other implementation of the ns-3 or recovering all the simulation with a proper interpolation middle-ware. The user's preferences or the requirements will guide the developers to choose the approach most interesting to them as seen in 4.2.

The easiest configuration of a co-simulator that includes several continuous-time domain-specific simulators is the use of a particular Tstep for all the simulators. The particular value of Tstep will greatly affect the validity of the results obtained and the length of the simulation because the greater Tstep is the less precision the simulation has but the less time the simulation takes. The discussion of Tstep needs to be addressed at implementation and production time.

STEP #4. Design of the user interface and results presentation

This is the last step of the methodology where the interface is designed. The co-simulator focuses on large facilities as commented in CHAR#8, this means that the users need no extra tools to design their own scenarios but layouts of typical facilities such as stadiums, or colleges will be preconfigured in the simulator.

There is no need to control the progress of the simulation step by step as commented in CHAR#2 so the focus will be on global trends with several simulations executed sequentially so as to get significant statistical results. To control the flow of this simulations a textual interface will be required. The results of the simulations need several visualizations as commented in CHAR#6 and additionally users need to process key aggregated values. To sum up, post-mortem results presentation is the most appropriate one. Figures 9.9, 9.10, 9.11 and 9.12 show the interfaces commented here.

Experiment Description

To assess the adequacy of the methodology proposed an experimental validation was carried out with the goal of verifying that the proposed methodology produces the most satisfactory co-simulator design for a particular application and to additionally test the operational limits.

HYDRA Home About Contact

Configuration

MASON parameters

MASON URL : /

NS3 parameters

NS3 URL : /

Scene parameters

Number of Users

Dimensions

Random Multiplier

Batch simulation parameters

Number of Steps

Number of Simulations

Figure 9.9: Basic configuration simulator interface

HYDRA Home About Contact

Sending message: [{"data": [{"x": 50.3, "y": 49.89}, {"x": 50.32, "y": 49.89}, {"x": 49.99, "y": 49.96}, {"x": 49.53, "y": 49.65}, {"x": 50.13, "y": 50.13}, {"x": 49.89, "y": 50.48}, {"x": 50.46, "y": 50.11}, {"x": 49.8, "y": 50.39}, {"x": 49.84, "y": 49.75}, {"x": 49.63, "y": 49.61}, {"x": 50.3, "y": 49.61}, {"x": 50.42, "y": 49.77}, {"x": 49.82, "y": 50.09}, {"x": 49.82, "y": 49.82}, {"x": 49.51, "y": 49.98}, {"x": 50.44, "y": 49.98}, {"x": 49.58, "y": 49.64}, {"x": 49.95, "y": 50.01}, {"x": 49.82, "y": 49.76}, {"x": 50.07, "y": 49.74}, {"x": 50.07, "y": 49.97}, {"x": 49.8, "y": 50.26}, {"x": 50.04, "y": 50.02}, {"x": 49.62, "y": 50.39}, {"x": 50.28, "y": 49.55}, {"x": 49.55, "y": 49.55}, {"x": 49.77, "y": 50.37}, {"x": 49.51, "y": 50.44}, {"x": 50.22, "y": 50.15}, {"x": 50.05, "y": 50.11}, {"x": 49.86, "y": 49.71}, {"x": 50.33, "y": 50.28}, {"x": 49.64, "y": 50.28}, {"x": 50.09, "y": 49.88}, {"x": 50.04, "y": 49.77}, {"x": 50.14, "y": 50.18}, {"x": 49.77, "y": 50.05}, {"x": 50.32, "y": 50.27}, {"x": 49.72, "y": 49.81}, {"x": 49.93, "y": 49.81}, {"x": 49.62, "y": 50.11}, {"x": 50.08, "y": 49.59}, {"x": 49.62, "y": 49.85}, {"x": 49.7, "y": 49.54}, {"x": 49.51, "y": 50.37}, {"x": 50.07, "y": 50.45}, {"x": 50.49, "y": 50.37}, {"x": 49.82, "y": 50.35}, {"x": 49.9, "y": 49.69}, {"x": 49.85, "y": 50.47}, {"x": 10, "y": 10}], "cmd": "INFO"}]

Simulation started

New message received:

[{"cmd": "START", "data": {"numUsers": 50, "dimension": 100, "randMult": 20}}]

Sending message: [{"data": [{"x": 50.47, "y": 49.61}, {"x": 49.96, "y": 50.11}, {"x": 50.28, "y": 50.35}, {"x": 50.09, "y": 49.71}, {"x": 50.17, "y": 50.46}, {"x": 49.98, "y": 49.77}, {"x": 50.33, "y": 50.4}, {"x": 49.75, "y": 50.02}, {"x": 50.49, "y": 49.54}, {"x": 50.25, "y": 50.23}, {"x": 49.63, "y": 50.1}, {"x": 49.52, "y": 49.9}, {"x": 50.06, "y": 49.62}, {"x": 50.13, "y": 49.98}, {"x": 50.04, "y": 50.03}, {"x": 50.18, "y": 49.56}, {"x": 49.66, "y": 49.87}, {"x": 49.5, "y": 50.1}, {"x": 50.15, "y": 50.45}, {"x": 49.66, "y": 50.39}, {"x": 49.55, "y": 49.89}, {"x": 50.09, "y": 50.34}, {"x": 49.89, "y": 50.48}, {"x": 49.76, "y": 50.43}, {"x": 49.8, "y": 49.88}, {"x": 49.89, "y": 50.38}, {"x": 49.81, "y": 50.12}, {"x": 50.13, "y": 50.05}, {"x": 49.85, "y": 49.51}, {"x": 49.74, "y": 49.68}, {"x": 50.09, "y": 50.13}, {"x": 50.1, "y": 50.14}, {"x": 50.1, "y": 49.77}, {"x": 49.82, "y": 50.03}, {"x": 49.88, "y": 49.64}, {"x": 50.17, "y": 50.46}, {"x": 49.69, "y": 50.32}, {"x": 50.36, "y": 50.11}, {"x": 50.27, "y": 49.93}, {"x": 49.6, "y": 49.69}, {"x": 50.26, "y": 49.79}, {"x": 50.41, "y": 50.24}, {"x": 50.03, "y": 50.42}, {"x": 50.2, "y": 49.82}, {"x": 50.47, "y": 49.91}, {"x": 49.57, "y": 49.68}, {"x": 49.8, "y": 49.62}, {"x": 49.98, "y": 50.25}, {"x": 50.47, "y": 50.28}, {"x": 49.67, "y": 50.34}, {"x": 10, "y": 10}], "cmd": "INFO"}]

Simulation started

New message received:

[{"cmd": "START", "data": {"numUsers": 50, "dimension": 100, "randMult": 20}}]

Search Go

Figure 9.10: Simulation execution and control interface

HYDRA Home About Contact

Select scene

Scene	Description	Edit
Scene 0	Several users with mobile phones move around a router, connecting and disconnecting from the WiFi.	
Scene 1	Sensing environment	
Scene 2	Building B at the ETSIT	
Scene 3	People moving in a classroom building	

Figure 9.11: Simulation design interface

To address this the previously defined co-simulator was implemented and three experiments were executed:

Figure 9.12: Results presentation temporal interface

- In **experiment#1**, a crowd simulation expert evaluates several quality parameters from the implemented co-simulator extracted in the step zero of the methodology. This experiment is a global evaluation of the co-simulator but concentrated on the first, second and fourth steps of the methodology in addition to the final user's requirements.
- **Experiment#2** is centered on step three where temporal measurements are calculated in several scenarios.
- **Experiment#3** evaluates the operation limits of the co-simulator and particularly scalability and accuracy.

Non-Methodological Co-Simulator Implementation

In the three experiments carried out, the values obtained are compared with the results from other three co-simulators implemented but these are not based on the proposed methodology, they are based on existing proposals related to this topic. These are the characteristics of these non-methodological co-simulators.

The first non-methodological co-simulator, **simulator#1**, was implemented based on a type of choreographed co-simulation paradigm where all the domain-specific simulations are gathered in a middle-ware that can communicate with the other simulators. The simulation models were kept as defined by default in the domain-specific simulators and no extra interface was

implemented, the graphic results were shown using the tools than ns-3 provides. In this co-simulators the domain-specific simulators run in parallel as can be seen in figure 9.13. In this case all the domain-specific simulators process independently a first approach to the scenario situation in the following step and then it is shared among the other tools. Processing this new data, the domain-specific simulators change their first calculations sharing and updating the process repeatedly until the data converges to a stable situation.

Figure 9.13: Simulator#1 architecture

Figure 9.14: Simulator#2 architecture

The second non-methodological co-simulator, **simulator#2**, was programmed based on a popular scheme for initial concept proofs in research. It is alike an "orchestrated by a one of the simulators" co-simulation paradigm, in which the most relevant domain-specific tool coordinates all the simulation [Kachan, 2010]. Similar to the original work, MATLAB is selected as the main simulator. And similar to simulator#1 the simulation models are kept in their default definition in the domain-specific simulators and no extra interface was developed. The coordination mechanism used was "stop-and-wait" and the results are provided using the tools that MATLAB offers. The architecture of this co-simulator can be seen in figure 9.14.

The third non-methodological co-simulator, **simulator#3**, is very similar to the one use based in the methodology with the difference being that the coordination mechanism used is parallel execution rather than stop-and-wait. With this architecture the performance of this co-

simulator can be easily compared with the one used in the methodology, particularly step three.

A benchmark simulator is assessed to enable comparisons between cutting edge tools and the proposed application-specific co-simulators. There is no simulator proposed in the state of the art that enables simulating CPSS taking into account all the subsystems equally as commented in chapter 2 so as an example if the simulation of a fire is necessary it requires a realistic fire expansion through the building but additionally the reaction of the people with the fire needs to be considered as well as possible interferences in wireless communications. This is why benchmark domain particular simulators usually achieve a better performance, meaning that they are faster and more scalable, and also the simulations are more simple. To set a reference a benchmark simulator centered on the simulation of the chosen scenario has been selected, as seen in section 9.2.1. Based on this, the social simulator Multi-agent system simulation of indoor scenarios (MASSIS) has been deployed and used [Pax and Pavón, 2015][Pax et al., 2017]. MASSIS relies on MASON, like the proposed co-simulator, but adds new interfaces and functionalities to simulate characteristics from the cyber and the physical worlds.

Detailed Description of Experiments

The assessment of the performance and adequacy of the proposed co-simulator was carried out by an expert on crowd simulations following a collection of quality parameters. In addition, the same evaluation was performed with the non-methodological co-simulators “simulator#1” and “simulator#2” and MASSIS simulator. A list of quality parameters was chosen to represent the adequacy of the obtained co-simulator for the scenario and also the adaptation of the three non-methodological co-simulators with the final user’s requirements is assessed. The chosen parameters were: usability by crowd management experts, easiness to include new kind of devices, scalability, suitability of the simulation model, accuracy of the simulation, customization and relevance of the final results.

Experiment#1 cannot evaluate the adequacy of the coordination mechanism because it is a technical solution that is related to the developers knowledge of the implementation of the co-simulator.

In **experiment#2** a validation was performed to assess if the methodological co-simulator implements the best coordination mechanism. To achieve this the required simulation time is evaluated and compared with simulator#3. All results have been compared and evaluated taking into account the development time spent in each system. This validation consisted on the definition of specific simulations that were carried out by all the simulators. Finally, as commented in previous sections, all the designed co-simulators should meet the final users' requirements. Anyway, after some time, the use of the originally designed co-simulator diverges from the original design but if this happens it is relevant to check if the tool still meets the final users' requirements with the changes applied. The results of this experiences were compared with "simulator#1", "simulator#2" and "simulator#3".

Experiment#3 evaluates the scalability of all the simulators based on the quantity of agents. To achieve this, the number of agents is progressively increased for a particular scenario until the tool is unable to perform the simulation and then the data generated is stored. This experiments also repeats the evaluation with a specific number of agents but increasing the quantity of parameters each agent has, evaluating the complexity of the agent model. Similarly to the first evaluation, several executions are implemented. Finally this experiment also tests the accuracy of the co-simulator where a crowd expert executes particular simulations with independent techniques. The data obtained is set as a reference and after this several simulations are executed modifying the parameter Tstep. The differences between both are compared and it determines the accuracy of the co-simulator. MASSIS defines entities with a fixed number of parameters and Tstep cannot be modified in an easy way so this benchmark simulator was only considered for the first part of the experiment. It should be noted that the results gathered cannot be compared between them because the complexity of the simulators is not the same, but the measures can be used as a references in the discussions.

Simulation Scenario

To execute the experiments previously described a specific simulation scenario has been defined. An interesting topic today is the suitability of public infrastructures in an emergency situation where an evacuation is required and specific situations need to be avoided such as human

avalanches or bottlenecks near the exits. Communications networks, particularly proprietary Wi-Fi networks, endure high stress and depending on the specific situation they may get isolated and disconnected from the network. Additionally in these events people behave in an unpredictable way what implies that common emergency systems such as alarms or panels may not be as effective as expected. The three worlds, the physical one, the cyber and the social is very interconnected in these kind of scenarios so the objective of the planned simulations is to assess the proposed methodology when creating a tool for a realistic CPSS application.

The scenario simulated is a representation of a base floor in one of the buildings at UPM full of classrooms. Some people move around this environment with their smartphones connected to the Wi-Fi network in the building and interacting with other people around. The simulation includes several corridors, eleven classrooms and some other spaces such as halls, bathrooms or offices. There is a maximum capacity of 800 people in the environment. The layout of the floor can be seen in figure 9.15.

Figure 9.15: Simulation scenario

A fire was simulated in that scenario and then the evacuation began. The increase of temperature and other information gathered from the emergencies system forced people to move

and interact with others causing the quality of the communications to decrease. Smartphones gathered the social information and sent the data to a central server deployed in a computer near the simulated environment. With the data obtained the location of groups of people is obtained, evacuation plans can be specified for the particular situation, the data can be communicated, etc. The simulation computes the evolution of the environment in the first ten minutes.

In experiment#1 the expert executed several simulations and in each one they modified the parameters considered relevant. In experiment#2 the simulation was executed several times each one with an increasing number of people in the environment. It was possible to assess the differences in the time developing the methodological co-simulator and simulator#3 that uses parallel execution. In experiment#3 the simulation was repeated varying certain parameters such as the number of agents, the quantity of parameters in an agent and the parameter Tstep. In each case fifteen simulations were executed calculating the result as the average value. To simulate each of the three sub-systems (cyber, physical and social) libraries and proposals have been used. Cellular network and Wi-Fi has been simulated with existing ns-3 libraries. The social environment has been implemented with MASON tools and emergency simulations proposals [Ren et al., 2009]. The main idea is employing inertial sensors available in the smartphones to predict the people behavior in the emergency checking the emotional state of the people. Here participative sensing makes no sense with passive monitoring being the most relevant approach. Sensors outputs are emulated with the coordination of social simulation, network simulation and physical simulation. The physical domain is assessed with numerical models for fire propagation [Kodur and Dwaikat, 2008].

9.2.2 Experimental Validation: Results

This section presents the results of the three experiments described in the previous section.

Results of Experiment#1

Table 9.4 presents the conclusions of the crowd simulation expert in experiment#1 where the quality of the methodological co-simulator, “simulator#1” and “simulator#2” are shown. The methodological co-simulator is globally better than the others and the benchmark simulator for the specific application. This means that the methodology achieves the objective of creating the best co-simulator for a particular application of CPSS. Each of the quality parameters is described in detail later.

Table 9.4: Quality parameters results in Experiment#1

Quality parameter	Methodological	Sim#1	Sim#2	MASSIS
Usability by crowd management experts	8	5	6	9
Easiness to include new types of devices	9	7	6	6
Scalability to advance scenarios	9	6	9	7
Adequacy of the simulation model	8	8	8	7
Accuracy of the simulations	7	8	8	8
Customization options	6	8	8	7
Interest of the results shown	9	7	7	8
Total	8	7	7.4	7.5

The expert took into account that all the co-simulators require the users specific knowledge from the cyber world, something not ideal. In the methodological co-simulator, the third-party engine hides most of the technical details that need to be controlled by the user in the secondary co-simulators. MASSIS simulator is the best because it includes a graphic interface, something very convenient to crowd simulation researchers.

Additionally, the third-party engine handles an easy way to extend the proposed co-simulator when new devices or domains need to be added. On the other hand, “simulator#1” needs the implementation of a middleware when a new domain-specific co-simulator is added, something complicated to non-technical users. “Simulator#2” has similar problems because all new devices require an equivalent description in MATLAB something that is usually unknown to the social experts. MASSIS simulator offers an embedded library with some existing devices but is hard to extend because it requires a new development and technological knowledge.

Related to scalability, the expert thinks that the methodological co-simulator addresses all the simulation scenarios in crowd management systems verification. This also happens in "simulator#2" because it is a specific property of "stop-and-wait" coordination mechanism as seen later. On the contrary, "simulator#1" has many difficulties with particular scenarios because it needs to manage a lot of signalization and this decreases the simulation performance. MASSIS simulator includes a lot of 3D interface, something that provokes scalability problems in simulations with a lot of elements.

The expert commented that the models were satisfactory in all the cases despite some improvements can be done such as the extension of the social aspects. Experts also commented that the simulation model included in MASSIS has some deficiencies with mobile entities because as an example, they don't correct the itinerary if a wrong path is followed.

Accuracy in all the co-simulators is reasonable but non-methodological co-simulators in addition to MASSIS enable users the fine tuning of this parameter because the third party engine hides some of the characteristics. Related to customization, the non-methodological co-simulators and MASSIS enable users the modification of the model, simulation routines and other aspects more easily than the methodological co-simulator, again due to the third-party engine.

The final quality parameter, the results presentation in the methodological co-simulator was really useful for the crowd expert but the other co-simulators require additional functionalities such as for example advanced animations. MASSIS includes an interesting graphic interface but with the problem that some representations are hard to get.

Overall, as previously commented, the proposed methodology enable the implementation of the most relevant co-simulator.

Results of Experiment#2

The results obtained in the second experiment can be seen in figure 9.16. It shows displays the average simulation time based on the number of people and smartphones, the agents. Both the results from the methodological co-simulator and the non-methodological ones. As commented

previously the maximum capacity is 800 people but an extra capacity of around 20% is studied taking into account up to 1000 people and 1000 smartphones.

Figure 9.16: Distributions of services quality in Experiment#2

The simulation time in figure 9.16 is represented in a logarithmic scale to be able to see all the values despite the several orders of magnitude of difference. The simulation time is higher for both co-simulators when the quantity of people is increased in more or less a complexity order $O(n^2)$. As expected, the methodological co-simulator that uses a stop-and-wait mechanism takes longer to execute the simulations than the non-methodological one that uses parallel execution.

The total time required to implement the methodological co-simulator is in the order of 1000 h while the identical simulator with parallel execution takes five times more, 5000 h. Considering this and the results presented in figure 9.16, figure 9.17 represents the function $S(n)$.

In the worst case presented in figure 9.17, more than a hundred simulations need to be executed to compensate the time spend developing the parallel execution mechanism but considering the simplest scenario ten million simulations are required. This is a huge number of simulations so the conclusion is that stop-and-wait is a suitable mechanism as indicated by the methodology.

Figure 9.17: Distributions of services quality

Results of Experiment#3

After the performance of the methodology has been assessed it is necessary to test the operational limits of the co-simulator.

The results related to the scalability based on the quantity of agents in each simulation scenario can be seen in figure 9.18. All the simulation that include 5000 agents or less performs well in the methodological co-simulator. When this number increases, the simulation algorithm collapses and sometimes the simulations can't be executed. Specifically with a number of agents of 50 thousands there is around a 50% chance of failure. A simulation is considered a fail when the coordination entity cannot communicate with one or more domain-specific simulator inside the co-simulator. Additionally simulations are blocked when the process that updates one of the domain-specific simulator cannot end before a Linux keepalive probe. These problems may appear because of hardware limitations, software coordination errors or communications stack overflows when the time for a simulator to update each Tstep is greater than a particular limit. The number of agents that can be considered when no problems arise also depends in the physical machine hosting the simulation. For this tests a 64-bit Linux Ubuntu 16 operating system with an Intel i5 processor and 8GB of RAM was used. This hardware configuration is

not designed for heavy simulations so the results gathered are not much agents per simulation. But because the same hardware is used to measure all the experiments the measurements obtained are suitable.

More than 80 thousand agents in the simulator doesn't work as can be observed in figure 9.18 with the limitation being the third party engine because it acts as an orchestrator that needs to process all the information. This creates a bottleneck that halts the simulation even if all the simulators are executing perfectly.

Figure 9.18: Distributions of services quality

The results are worse in "simulator#2" and "simulator#3". In "simulator#2" the orchestrator, MATLAB, needs to execute their own simulations in addition to the orchestration, so the bottleneck happens at around seven thousand agents with a 50% error. "Simulator#3", with a parallel execution scheme, function poorly with a 60% error chance at around five thousand agents because the data is not generated in an ordered way.

When choreographed simulation is used the situation improves as can be seen for "simulator#1". It can hold a greater number of agents because it has a distributed management system. "Simulator#1" halts when it is unable to manage the signaling load among the simulators. This happens at around seventy thousand agents with an error rate of 40%.

Despite being low numbers, the greatest public facilities can host up to 200,000 people. The

proposed scenario can host up to three thousand people, something that the proposed co-simulator can execute without problems. The designed co-simulator is not suitable for large-scale simulations that include hundreds of thousands of people so a different methodology would be necessary for this case. Additionally the hardware used is worst than other ones used in scalability studies [Lorig et al., 2015], what means that this results could be improved.

The benchmark simulator MASSIS enables larger simulations until errors arise because even MASON can be executed with up to a million agents [Lorig et al., 2015]. But the problem is that these simulations include only social aspects ignoring physical and cyber ones.

Another relevant variable in scalability is the quantity of parameters considered for each agent and the simulations were executed with 800 agents, as commented in section 9.2.1. The results gathered from this test can be observed in figure 9.19.

Figure 9.19: Distributions of services quality

The tendencies seen in figure 9.19 are similar to the ones seen in figure 9.18. This makes sense because simulation algorithms and signaling load is related similarly to the quantity of agents and their parameters. This means that the discussions related to figure 9.18 are valid here. To compare the results table 9.5 shows the most relevant characteristics of figure 9.19.

The methodological co-simulator achieves the final users' requirements as can be seen in figure

Table 9.5: Scalability of the proposed co-simulators

Key aspects	Methodological	Sim#1	Sim#2	Sim#3
Maximum parameters without failing	30	300	7	4
Parameters with a 50% fail	310	440	40	30
Maximum number of parameters	500	620	440	440

9.18 and figure 9.19. Specifically several tens of thousands agents can be simulated as indicated in REQ#3.

The results of the accuracy evaluation can be seen in figure 9.20. This takes into account a definition of global error to evaluate the accuracy of the proposed co-simulator when Tstep is reduced if possible. The aggregated absolute error in the simulations is the aggregated difference of probability at the final time and the reference parameter, and then normalized. The error committed in the methodological co-simulator is constant with a value of around 12% because Tstep is not controllable by final users as defined by CHAR#2 because the co-simulator is event-oriented.

Figure 9.20: Distributions of services quality

All of the non-methodological co-simulators enable the manipulation of Tstep and in general, for large values the error increases, because there are abrupt changes, and for lower values the error decreases as the simulation is more continuous. But as can be seen, in "simulator#2"

and "simulator#3" the error increases again then Tstep is less than a particular threshold of around 0.1 s. The explanation of this situation is that the algorithm blocks with a very small Tstep because the orchestration entities are too slow to update the global state and some data is lost what means an increase in error.

9.2.3 Discussion

Taking into account all the results commented previously, the correctness of the methodology proposed can be confirmed. The results from experiment#1 assure that the selected co-simulation paradigm, the designed simulation model and lifecycle and the presentation interface are the most significant ones. Experiment#2 proves that the designed simulation tool is the most proper related to the time and the precision of step three is very good. The other steps are subjective to the users' needs and impressions, but an expert has analyzed them to reduce the human factors whenever possible.

The proposed co-simulator has a wide operation range and supports several tens of thousands of agents and up to five hundred parameters for each agent. This means that final users' requirements are fulfilled and the methodology proposed generates the most adequate tool for each application. But the co-simulators can be extended in the future and this is not taken into account in the proposed methodology. If this happens business or corporative arguments are the main consideration to be done.

The proposed methodology is but the initial stage in the co-simulator creation process because then implementation needs to be done and users need to be capable to generate adequate simulations. All this presents new challenges and problems that are not the focus of this section but could influence in the final results.

9.3 Analysis of the integration of a simulator into an AmI environment to predict future events

Having discussed about the viability of the integration of the simulator into an AmI environment in chapter 7, in this section I get some numbers for specific simulation of scenarios to get an idea of under what circumstances the simulation is viable and under what ones it is not viable.

The **first scenario** is just a single room with people inside walking with their mobile phones connecting and disconnecting from several Wi-Fi hot-spots and approaching them when necessary. In this scenario I have a fixed number of people since the beginning of the simulation and will stay constant all over it. The simulation will end after a predefined amount of simulation time but for the purposes of the analysis and considering the scope I will leave this time at 5 minutes. I will increase the number of people in the scenario to study how the times vary depending on the number of elements in the simulation.

The **second scenario** is a bit more complex where there is people entering a building at a fixed ratio and deciding whether to take the stairs to go to a higher floor or to take the lift. Once people get to the lift or the stairs they will leave the scenario. There is no people initially at the main hall but people will get inside over time. In addition to the people there are several hot-spots that give Wi-Fi coverage in the area and several sensors that monitor the people and other elements in the scenario such as temperature or presence. I will vary the ratio of people entering the building in the scenario again to study how the times vary as more elements are present in the simulation.

In all this scenarios I will use a fixed Tstep of 1 second that as explained in this chapter it defines the simulation time that spans in a simulation cycle running both the social simulation and the network simulation for that amount of simulation time.

During the simulation process I will calculate the time it takes to start the simulation, to process each simulation cycle and finally to end the simulation repeating this measurements for each three scenarios and with different orders of magnitude of people varying the parameters

described before, starting with 10, then 100 and finally 1000. The simulation has been executed in a machine with an Intel Core i7-7700 processor at 3.6 GHz and 16GB of RAM.

Figure 9.21: Graph with times from scenario 1

The resulting times from executing the **simulation of scenario 1** can be seen in figure 9.21. It can be seen that the starting time of the simulation increases linearly with the number of people taking a whole second with a scenario of 1000 people. Similarly the time it takes to complete a cycle increases linearly with the number of people moving in the scenario taking more than a second to execute one second of the simulation time making this simulation completely useless with the simulator integrated into an AmI environment as it takes more time to simulate than the time it simulates. Finally the ending time remains constant with the number of people as it only ends the simulation in both the social simulator and the network simulator but generates no additional data.

The conclusions gathered from the analysis of the scenario 1 are that, for 100 people or less moving inside the room, the simulation can work very good inside an AmI environment as the simulator goes around 7 times faster than the simulation time with a reasonable starting and ending time. On the other hand, with much more than 100 people, the simulation starts going really slow and becomes unfeasible to integrate it with an AmI environment but otherwise it

could still be viable as an offline analysis tool.

Figure 9.22: Graph with times from scenario 2

The resulting times from executing the **simulation of scenario 2** can be seen in figure 9.22. It can be seen that the starting time of the simulation remains constant with the number of people, as expected, because there is no initial people in the environment as they will start entering it over time. In addition it can be observed that the time it takes to process a simulation cycle increases linearly with the number of people getting inside the building, getting up to nearly 600 ms with 1000 people per minute. This value is still lower than the 1000 ms marked by the `tstep` but will leave little time to react. Take into account that a 5 minutes simulation would take around 3 minutes to complete, enough time for the environment to have change considerably. Finally, similarly to scenario 1, the ending time remains constant with the number of people as it only ends the simulation in both the social simulator and the network simulator but generates no additional data.

The conclusions gathered from the analysis of the scenario 2 are that, for 100 people per minute or less entering the building, the simulation can work very good inside an AmI environment as the simulator goes around 7 times faster than the simulation time with a reasonable starting and ending time. On the other hand, with much more than 100 people per minute, the simulation

starts going really slow and becomes unfeasible to integrate it with an AmI environment but otherwise it could still be viable as an offline analysis tool.

9.4 Use of prosumerization for the simulation of Ambient Intelligence environments

In order to guide this PhD Thesis work I will use the three research questions presented in the introduction. An experimental validation was carried out in order to address these research questions. During the experimentation step, it was proposed to prosumers the creation of four different services to solve four specific problems using the prosumer framework DataQuest. Then it was analyzed the quality of the services created by the prosumers using DataQuest and several specialists developed these same four services in order to get the time necessary to program this services without using a prosumer environment.

9.4.1 Context

Eighteen people (hereafter, participants) at UPM carried out the empirical study using DataQuest. These people are first year telecommunications engineering students and were collaborating with the GISAI group at UPM; creating four different services. To evaluate whether prosumers were efficient creating the services, the experts reviewed each service created by the participants, testing them in a simulated environment created for this experiment. A full factorial experimental design was followed where all 18 participants created four different services.

In addition the same four services were presented to twelve developers that programmed the services and the time spent to create the service was measured. All the expert programmers are master students at UPM collaborating with the GISAI group at UPM.

9.4.2 Planning

The experiment was divided into three phases:

- Training phase: The participants were trained in the principles of service creation using a prosumer environment using DataQuest. The training sessions consisted of 2 hours of theory at the beginning of the experiment execution and personal interviews to solve specific issues raised by the participants during the services creation. In addition the expert developers were trained during 2 hours how to access the semantic platform FIESTA-IoT ontologies in order to access the information from the sensors.
- Service creation phase: During this phase, each participant had to create four services for solving the four problems proposed by the experts in random order and each developer had to program the service assigned to him. These services were:
 - Generate a dataset with all the observations from today from two different devices.
 - Generate a dataset with the observations from the third of December of a pressure and a humidity sensor, adding a value deviation of 5%.
 - Add a real time service that monitors three different temperature sensors softening the value between 10 and 30 degrees Celsius.
 - Add a real-time service where a humidity sensor is monitored softening the values between 10 and 90%, a temperature sensor changing the units from Celsius to Fahrenheit (multiply by 1.8 and add 32) and a pressure sensor adding a 2% value deviation and also normalizing the frequency of the three sensors to one measurement per minute.
- Services evaluation phase: All the services created by the participants were evaluated by the experts who assigned a quality score to each one. In this phase the participants also assessed DataQuest by means of a survey. The time spent by the developers creating the services was measured.

9.4.3 Data Collection

To address the first research question it was necessary to conduct a study on the quality of the services created by the prosumers. The experts managing the experiment evaluated all the services created using DataQuest and assessed the quality of the services elements used by the participants to build each service. The quality evaluations were performed after the services prepared by the participants were delivered.

- The time spent by participants to create the services. The experts carrying out the experiment analyzed the log files created by DataQuest when the participants created the services with a semiautomatic tool in order to determine the building time of each service.
- The time required to create the services by the developers.

The data collected to address the third research objective were the subjective satisfaction evaluation of the participants obtained through an anonymous survey filled up at the services evaluation phase by the participants.

9.4.4 Result

This section presents and discusses the results obtained in the experimental validation and will be presented according to the research questions proposed in this work.

Analysis of: What is the quality of the services developed by prosumers?

Table 9.6 summarizes the average quality of the services built by the prosumers during the "service creation phase", ranked from 0 to 10 with 0 meaning a bad service, probably unfinished and a 10 meaning a perfect service.

The results obtained indicate that the global quality of services built were good but depended on the difficulty of the requirements for each service, the first two services were easy ones and

Table 9.6: Summary of the quality of the service built

	Ser. #1		Ser. #2		Ser. #3		Ser. #4		Total	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
Problem resolution	4.83	2.04	5.33	1.81	5.94	2.1	3.22	1.86	4.83	2.17
Service accuracy	6.33	1.91	4.89	2.56	4	2.95	3.72	2.54	4.74	2.67
Test results	5.89	2	4.83	1.89	4.94	2.82	4.06	2.39	4.93	2.35
Feedback	7.33	1.75	6.5	2.18	5.61	2.81	4.56	2.83	6	2.6
Quality aggregate	6.1	1.05	5.39	0.89	5.12	1.77	3.89	2	5.12	1.67

in general scored good but in the case of the third and fourth services, low quality scores were obtained. In addition the first two services are to create datasets while the last two are to create a service that feeds the simulator in real-time, so it could be stated that people understood better how to create datasets but had more problems creating a real-time service. A special note should be taken with the fourth service, as this is the most complex and it seems that participants have had a difficult time with it getting confused with what to do or how to select the appropriate actions in order to get the desired service.

Figure 9.23: Distributions of services quality

Figure 9.23 shows the distributions of the quality aggregate scores for the total of the services and also for each one. The first service obtained the best scores for quality, with more than 83% of the services above 5 and a narrow value for the standard deviation; meaning that the quality is very predictable and it is easy to obtain in a prosumer environment.

Table 9.7: Summary of the time spent in creating the services

	Domain experts				Developers			
	Min	Max	Median	SD	Min	Max	Median	SD
Service #1	4	11	9	2.09	17	32	28	4.65
Service #2	9	18	10.5	2.98	17	38	27	6.72
Service #3	11	35	28	7.07	15	35	28	7.07
Service #4	13	42	25	8.37	12	36	32	8.95
Total aggregated	4	42	13.5	7.89	12	38	28.5	6.84

As shown in Figure 9.23 the standard deviation for the second service is very narrow (0.89), while the standard deviation for the fourth one is very wide (2). This means that quality for the second service is more predictable than the first service; the quality is easily obtained with a prosumer environment whenever the difficulty is accessible to the prosumers. The fourth service got lower scores on quality, only 33.33% of the services exceeded 5 points. The quality scores were very low for this service because it is the most difficult to build. As stated above, in a prosumer environment, the prosumers can achieve high quality products as long as it is not too difficult for them, but it starts to be less useful as more complex services are required.

Analysis of: Who are more efficient: prosumers or expert developers?

It was carried out an analysis of the time necessary to create a service using a prosumer environment, studying the time spent by the prosumers using DataQuest and comparing it with the time spent by code development experts in a traditional environment, both with the same requirements.

A summary of the time (measured in minutes) necessary to build and develop services is shown in table 9.7. It is shown the values of the mean, median and standard deviation for each service and for both groups.

As explained in section 9.4.3, the prosumers took two hours of training; this training time has not been taken into consideration to measure services building time. Similarly the expert developers also received two hours of training that has not been taken into account to measure the services building time. In other words, the time measured for the prosumers and for the expert developers was the time taken to build each service since they were presented the task until they completed the service.

The prosumers took considerably less time to build services. Moreover, the building time increases slightly depending on the difficulty of the service. However, the development time that developers took remains practically constant in the four services, and the same happens with the standard deviation. This means that the difficulty affects the prosumers as they are not professional developers, while a slight increase in the difficulty of the services does not require more time for developers as the inclusion of some more conditions or operations is not comparable with having to configure a new rule with different actions and filters. A Mann-Whitney U test was conducted to confirm that the prosumers take less time to build a service. The results are positive and support the previous assertion. The results of the test were within the expected significance, $\rho < 0.001$

Figure 9.24 shows the distribution of the time spent in creating the services by prosumers and the experts developers.

Figure 9.24: Distributions of time spent in creating the services

It can be stated that the prosumers took less time to solve a problem by building a service in a prosumer environment compared to professionals developing the same functionality in a traditional environment. As explained before, the time the prosumers and developers took to learn and gain experience in the use of their specific tools has not been taken into account.

However, the prosumers required 2 hours of training to learn and gain experience in the use of their specific tool. Developers also required 2 hours of training to learn the basics required to program the services but it took longer for them to learn and gain experience in software development tools.

Analysis of: How satisfied are prosumers using a prosumer platform?

This section describes the analysis of the satisfaction perceived by the users. In this experiments there were two kinds of users: participants and trainers. Participants were asked by trainers on two questions on the use of DataQuest: satisfaction and stress; while trainers were asked about satisfaction and relevance. The results of the trainers were not representative, as there was the cooperation of only three people; although they showed great satisfaction in its use.

Figure 9.25: Satisfaction evaluation

Figure 9.25 shows that the overall evaluation of the participants for satisfaction was positive, with a mean value of 4.2; but nevertheless there were high scores for the stress measurements. According to the opinions of their trainers, this is because the participants reacted badly to the pressure of the experimentation because during the experiment there were five people unknown to them. For future experiments the trainers suggested us to become familiar with them before running an experiment.

9.5 Conclusions

In this chapter I have presented the experimentation done to validate the goals and RQs proposed in section 1.3 that have been the foundation of this PhD Thesis work.

I have addressed **Goal 1:** *To obtain a more complete simulation of AmI environments using Hydra than executing a social simulator and a network simulator separately*, carrying out a case study where four experts have evaluated the use of Hydra compared to executing MASON and ns-3 simulators independently in three specific scenarios. Additionally twenty one learners have evaluated the satisfaction using Hydra simulator. The conclusion obtained is that the use of Hydra improves the analysis of AmI environments but the system still needs improvement.

I have addressed **Goal 2:** *To propose a methodology that will help in the creation of co-simulators for AmI environments*, evaluating the performance and the invested time developing a CPSS co-simulator using the methodology proposed in chapter 4. The conclusion is that the proposed methodology generates the most adequate tool for each application, specially in terms of cost and time, but it lacks in the possibilities of adding future extensions.

I have addressed **Goal 3:** *To predict future events in an AmI environment using Hydra*, carrying out an analysis of the simulator Hydra with different parameters in order to establish the boundaries of its integration into an AmI environment to predict future events. The conclusion is specific to the different scenarios obtaining the maximum number of agents for the particular scenario where Hydra is still viable in a AmI environment.

And finally I have addressed **Goal 4:** *To improve the user experience with Hydra applying prosumerization techniques*, where I have carried out a study of the integration of Hydra with a prosumer framework where non-expert developers have created four different services and the results have been compared with those of expert developers creating the same four services without the use of the prosumer framework. The conclusions are that the use of a prosumer framework improves the creation of these services due to the flexibility in it.

Chapter 10

Conclusions

This PhD thesis work arises from the lack of systems that helps in the study and deployment of AmI environments where one needs to focus on both the people that surround the environment and the devices and network that is integrated into the environment. In order to fulfill this task, I have proposed an AmI simulator, called *Hydra*, that integrates both a social simulator and a network simulator.

This thesis is embraced in the Advance Communications, Information Technology and Public Safety NIST areas and the Information Society science area by the European Commission.

In this chapter, I discuss the results obtained for the different goals I have proposed in this work and answer the RQs associated with each goal. The following are the goals stated in section 1.3.

- **Goal 1:** *To obtain a more complete simulation of AmI environments using Hydra than executing a social simulator and a network simulator separately.*
- **Goal 2:** *To propose a methodology that will help in the creation of co-simulators for AmI environments.*
- **Goal 3:** *To predict future events in an AmI environment using Hydra.*
- **Goal 4:** *To improve the user experience with Hydra applying prosumerization techniques.*

The contributions proposed in this PhD Thesis to validate each of the previous goals are analyzed in the following sections in this chapter.

10.1 *To obtain a more complete simulation of AmI environments using Hydra than executing a social simulator and a network simulator separately*

After considering the lack of an existing AmI simulator that supports both an in-depth analysis of the people and the devices existing in these kind of environments, I have analyzed the different features present in actual social and network simulators and selected the most relevant ones in order to obtain those needed in an AmI simulator in addition to some more that appear from the integration of both simulators. With these requirements, I have proposed an AmI simulator called *Hydra*, created from the integration of an existing social simulator, MASON, and an existing network simulator, ns-3.

Based on the requirements analyzed previously and other characteristics of the social and network simulators, I have proposed an architecture for an AmI co-simulator that integrates both a social simulator and a network simulator taking into account their different nature. In order to support this architecture, I have also defined a data model, several sequence model and a time model, based on the necessities of the system.

In order to address Goal 1, I have answered the following RQs in chapter 3 supported by the experimentation discussed in section 9.1:

RQ1: Does the network and social co-simulation offer more features than the simple execution of each one separately?.

Based on the social simulator and network simulator features analyzed in section 2.4 and the Hydra features analyzed in chapter 3, an evaluation has been carried out by experts with the conclusion that the network and social co-simulator Hydra offers more features than the simple

execution of each one separately.

RQ2: Does the network and social co-simulation help in the analysis and drawn of conclusions?

As defined in section 9.1, a conclusion is something relevant extracted from the simulation in the expert's opinion. After doing the experimentation with experts simulating three different scenarios using Hydra and using MASON and ns-3 I concluded that more relevant conclusions can be provided for the simulation of AmI environments than simply using a social simulator and a network simulator.

RQ3: What is the perception in the use of the co-simulator?

An experimentation with twenty-one students who had to simulate AmI scenarios using *Hydra* has been carried out. They were taught to use the system, then they used Hydra to simulate several scenarios and finally were surveyed about their satisfaction using Hydra. Overall they were satisfied with the use of the system, except with the analysis step that still needs some work to be done.

10.2 *To propose a methodology that will help in the creation of co-simulators for AmI environments*

Based on the existing simulators and methodologies, I have proposed a methodology for the design of CPSS co-simulators. This methodology proposes four steps in order to fulfill the users' needs and requirements that include the selection of a paradigm, the particularization of the general model, a selection of the coordination mechanism and the design of the interface.

In order to address Goal 2 I have answered the following RQ in chapter 4 supported by the experimentation discussed in section 9.2:

RQ4: Does a methodology help in the creation of AmI co-simulators?

In order to answer this research question, a co-simulator was created using the methodology proposed and then several experiments were carried out to evaluate the different steps in the

methodology. The first addresses steps one, two and four of the methodology, in it various quality parameters were evaluated by a crowd simulation expert. The second experiment is focused on the third step, in the experiment temporal measurements about the required simulation time in different simulations are done. The third experiment evaluates the operation limits of the generated system, particularly scalability and accuracy. This experiments validated the proposed methodology with positive results.

10.3 To predict future events in an AmI environment using Hydra

In addition to using a simulator to predict the behavior of the different elements in a predefined environment, it can be integrated into a real environment receiving real data from it and giving feedback to it if possible usually controlling the actuators in the environment. I have integrated our co-simulator Hydra in an AmI environment obtaining data from sensors and actuators that exist within it. The goal of this integration is the prediction of a near future state of the environment based on all the data gathered and the reaction of the system to any problem that may arise. Based on the use case of the environment and the limitations, the system can inform about the environment not working properly and act accordingly using the available actuators.

In order to address Goal 3, I have answered the following RQ in chapter 7 supported by the experimentation discussed in section 9.3:

RQ5: Does the integration of a simulator in an AmI environment help in the prediction of future events?

I have studied the limitations of Hydra depending on the characteristics of the environment, the number of elements inside it and the reaction time based on the use case. Using the three existing scenarios in the simulator as defined in section 9.2, each one has been executed with a varying number of agents from 1 to 1000 and the conclusion is that it is viable the integration of Hydra in scenarios where up to 100 people are moving around. With more than 100 agents each

second in real time takes more than one second to simulate, making the simulation non-viable to be used integrated into an AmI environment.

10.4 *To improve the user experience with Hydra applying prosumerization techniques*

One problem arises when using a system such as Hydra, and it is the necessity of an expert developer to improve the system when new functionalities are required but it is also necessary an expert in AmI environments to make use of it. This problem could be solved with the addition of a prosumer framework that enables an easy interaction with it and the possibility of expanding its use without the hassle of having to know how to program it.

I have added a prosumer framework to our co-simulator Hydra, that adds configuration capabilities to enable the simulation to gather data from a semantic platform and execute specific simulations in an environment based on that data using rules based on actions, conditions and actions.

In order to address Goal 4, I have answered the following RQs in chapter 8 supported by the experimentation discussed in section 9.4:

RQ3: What is the perception in the use of the co-simulator?

An experimentation was carried out with two kinds of users: participants that were asked about satisfaction and stress, and trainers that were asked about satisfaction and relevance. The overall evaluation was positive with the exception of the condition rule that was a little complex as explained in section 9.4. Based on the prosumer's opinion about the use of this framework it greatly improves the possibilities of the simulator without the need to know a programming language to change its behavior and that can be used by an expert in AmI environments.

RQ6: What is the quality of the services developed by prosumers?

A group of participants was taught to use Hydra to create services and these services were then evaluated by a group of experts. The overall quality of the services created was very good with the exception of the last one, the most complex one that had some complications, as seen in section 9.4.

RQ7: Who are more efficient: prosumers or expert developers?

In order to answer this question, an analysis of the time necessary using the prosumer environment was carried out, studying the time spent by prosumers and developers creating the services. The result of the experiment was that the domain experts took considerably less time creating simpler services but the time was nearly the same when creating complex ones.

Chapter 11

Future work

This work has presented a system integrating two existing simulators, MASON a social simulator and ns-3 a network simulator, to create an AmI simulator. But a lot of work remains to be done. During the different chapters in this PhD Thesis work I have proposed the foundations for some ideas but the developing of these contributions is beyond this work. Here I recollect them for easier access:

- **Improve the prediction in the integration of Hydra to predict future events:** In chapter 7 I presented different possibilities to study the prediction of future events in an AmI environment without particularizing one. A deeper study should be carried out where the best solution is chosen and integrated with the simulator.
- **Improve the rules used in the prosumer framework:** In the experiment carried out in chapter 8 users stated that the current state of the events, condition and actions used in the rules were somehow confusing. Another study should be done in order to improve them, maybe removing the condition as it proved to be the most confusing part.
- **Improvement of the Hydra GUI:** During the different experiments performed, users have encountered some problems while using the system as has been commented in chapter 9. Some of the screens in the interface need a restructuring of the elements making it more clear its purpose.

In addition to the ideas previously proposed, here I propose other ones to continue this work with future lines:

- **Integration of different social and network simulators:** In this work, I have integrated two particular simulators, a social simulator called MASON and a network simulator called ns-3 but given the modular structure of the architecture presented in this work other solutions should be studied that might improve any of the used simulators. The integration shouldn't be too hard mainly requiring the modification of the engine interface and the creation of a new adapter in the simulator chosen.
- **Integration of different simulators other than social and network ones:** Depending on the particular use case for an AmI environment other simulators might be used in addition to the social and network ones to improve the results in a particular case. Some examples could be the use of a fire simulator to better study the behavior of people in an emergency evacuation, a hardware simulator to get a more accurate simulation of the behavior of the devices or a weather simulator might be useful if our environment includes a courtyard or other similar outside room. Including this simulator in the simulation workflow shouldn't be too difficult requiring the creation of an adapter for the chosen simulator and a modification of the engine interface and the communication between the simulators.
- **Integration of external data sources:** Instead of generating the data inside the simulator, based on previous data, the simulator could obtain some or all of its data from external sources. This could be real devices, similar to what I have studied in chapter 7 but it could also be from existing sources such as the weather forecast from the Meteorological Agency, people movement from the Traffic Department or some specific data from the Statistics Office.

Publications

In this section I present the publications I have written during the time period of the realization of this PhD Thesis work. First appear the publications directly related to this work and then other articles indirectly related.

These scientific articles are indexed following the Journal Citation Reports (JCR) and are directly related to the work done in this PhD Thesis:

- Sánchez, B. B., Alcarria, R., Sánchez-Picot, Á., & Sánchez-de-Rivera, D. (2017). A Methodology for the Design of Application-Specific Cyber-Physical Social Sensing Co-Simulators. *Sensors*, 17(10), 2177. [JIF 2.475, Q2].
- Sánchez-Picot, Á., Martín, D., Bordel, B., & Sánchez-de-Rivera, D. (2018). AmI environments simulations approach integrating social and network aspects: A case study. *Journal of Ambient Intelligence and Smart Environments*, 10(4), 303-314. [JIF 0.878, Q4].

The following is an article in an International Conference indexed in the Computing Research and Education (CORE) association and directly related to the work done in this PhD Thesis:

- Bordel, B., Iturrioz, T., Alcarria, R., & Sanchez-Picot, A. (2018, June). Provision of next-generation personalized cyber-physical services. In *2018 13th Iberian Conference on Information Systems and Technologies (CISTI)* (pp. 1-6). IEEE.

The following are other articles in International Conferences directly related to the work done in this PhD Thesis:

- Sánchez-de-Rivera, D., Bordel, B., Sánchez-Picot, Á., Martín, D., Alcarria, R., & Robles, T. (2019, February). Sharing Device Resources in Heterogeneous CPS Using Unique Identifiers with Multi-site Systems Environments. In *International Conference on Information Technology & Systems* (pp. 153-164). Springer, Cham.
- Bordel, B., Alcarria, R., Sánchez-Picot, Á., & Sánchez-de-Rivera, D. (2019, February). Cyber-Physical Systems for Environment and People Monitoring in Large Facilities: A Study Case in Public Health. In *International Conference on Information Technology & Systems* (pp. 406-416). Springer, Cham.
- Sánchez-Picot, Á., Sánchez-de-Rivera, D., Robles, T., & Jiménez, J. (2018, June). Time Analysis of the Integration of Simulators for an AmI Environment. In *International Symposium on Distributed Computing and Artificial Intelligence* (pp. 157-164). Springer, Cham.
- Martín, D., Bordel, B., Alcarria, R., Sánchez-Picot, Á., de Rivera, D. S., & Robles, T. (2017, November). Prosumerization Approach to Semantic Ambient Intelligence Platforms. In *International Conference on Ubiquitous Computing and Ambient Intelligence* (pp. 109-120). Springer, Cham.
- Alcarria, R., de Andrés, D. M., Bordel, B., de Rivera, D. S., Sánchez-Picot, Á., & Robles, T. (2017, January). A service-oriented monitoring system based on rule evaluation for Home Automation. In *2017 IEEE International Conference on Consumer Electronics (ICCE)* (pp. 329-330). IEEE.
- Bordel, B., de Rivera, D. S., Sánchez-Picot, Á., & Robles, T. (2016). Physical processes control in industry 4.0-based systems: A focus on cyber-physical systems. In *Ubiquitous Computing and Ambient Intelligence* (pp. 257-262). Springer, Cham.
- Sánchez-Picot, Á., de Andrés, D. M., Sánchez, B. B., Garrido, R. P. A., de Rivera, D. S., & Robles, T. (2016). Towards a Simulation of AmI Environments Integrating Social and Network Simulations. In *AmILP@ ECAI*.

- Martín, D., Bordel, B., Alcarria, R., Sánchez-Picot, Á., de Rivera, D. S., & Robles, T. (2016, November). Improving learning tasks for mentally handicapped people using Aml environments based on cyber-physical systems. In *International Conference on Ubiquitous Computing and Ambient Intelligence* (pp. 166-177). Springer, Cham.
- de Rivera, D. S., Alcarria, R., Martín, D., Sánchez-Picot, Á., Bordel, B., & Robles, T. (2016, March). Distributed Query Results and IoT Data in a Publish-Subscribe Network Implementing User Notifications. In *2016 30th International Conference on Advanced Information Networking and Applications Workshops (WAINA)* (pp. 778-783). IEEE.
- Sánchez-Picot, Á., Martín, D., de Rivera, D. S., Bordel, B., & Robles, T. (2016, March). Modeling and simulation of interactions among people and devices in ambient intelligence environments. In *2016 30th International Conference on Advanced Information Networking and Applications Workshops (WAINA)* (pp. 784-789). IEEE.
- Martín, D., Alcarria, R., Sánchez-Picot, A., Robles, T., & de Rivera, D. S. (2014, December). A four-leaf clover shape methodology for prosumer service developments. In *International Conference on Ubiquitous Computing and Ambient Intelligence* (pp. 488-495). Springer, Cham.
- Martín, D., Alcarria, R., Sánchez-Picot, Á., & Robles, T. (2014, December). Decision Making Environment Based on Prosumer Services for Tracing Drugs in a Hospital Pharmacy Department. In *International Conference on Ubiquitous Computing and Ambient Intelligence* (pp. 324-332). Springer, Cham.

The following are other articles in international journals directly related to the work done in this PhD Thesis:

- Sánchez, B. B., Alcarria, R., de Rivera, D. S., & Sánchez-Picot, A. (2016). Enhancing Process Control in Industry 4.0 Scenarios using Cyber-Physical Systems. *JoWUA*, 7(4), 41-64.

- Martín, D., Alcarria, R., Sánchez-Picot, Á., & Robles, T. (2015). An ambient intelligence framework for end-user service provisioning in a hospital pharmacy: a case study. *Journal of medical systems*, 39(10), 116.

These scientific articles are indexed following the JCR and are indirectly related to the work done in this PhD Thesis:

- Bordel, B., Alcarria, R., Robles, T., & Sánchez-Picot, Á. (2018). Stochastic and information theory techniques to reduce large datasets and detect cyberattacks in Ambient Intelligence Environments. *IEEE Access*, 6, 34896-34910. [JIF 3.557, Q1].
- Sánchez, B. B., Alcarria, R., De Rivera, D. S., & Sánchez-Picot, A. (2016). Predictive algorithms for mobility and device lifecycle management in Cyber-Physical Systems. *EURASIP Journal on Wireless Communications and Networking*, 2016(1), 228. [JIF 1.529, Q3].
- Alcarria, R., Martín, D., Robles, T., & Sánchez-Picot, Á. (2016). Enabling Efficient Service Distribution using Process Model Transformations. *International Journal of Data Warehousing and Mining (IJDWM)*, 12(1), 1-19. [JIF 0.727, Q4].
- Bordel, B., Alcarria, R., Martín, D. & Sánchez-Picot, Á. (To be published). Trust provision in the Internet of Things using transversal blockchain networks. *Intelligent Automation and Soft Computing - An International Journal*, TSI Press. [JIF 0.652, Q4].
- Martín, D., Alcarria, R., Robles, T., & Sánchez-Picot, Á. (2016). Prosumer framework for knowledge management based on prosumer service patterns. *International Journal of Software Engineering and Knowledge Engineering*, 26(07), 1145-1173. [JIF 0.299, Q4].

The following are articles in International Conferences indirectly related to the work done in this PhD Thesis:

- Sánchez, B. B., Sánchez-Picot, Á., & De Rivera, D. S. (2015, July). Using 5G technologies in the Internet of things handovers, problems and challenges. In *2015 9th International*

Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (pp. 364-369). IEEE.

- Alcarria, R., Martín, D., Robles, T., & Sánchez-Picot, A. (2014, December). End-User Service Distribution for Efficient Inter-Domain Execution. In International Conference on Ubiquitous Computing and Ambient Intelligence (pp. 480-487). Springer, Cham.
- de Rivera, D. S., Sánchez-Picot, Á., Alcarria, R., de Andrés, D. M., & Robles, T. (2014, December). Smart dynamic pricing based on ECA rules and electronic ink labeling for retail. In International Conference on Ubiquitous Computing and Ambient Intelligence (pp. 468-475). Springer, Cham.
- Sánchez-Picot, Á., Robles, T., Alcarria, R., & Martín, D. (2014, December). Services Collaboration Supported by a B2B Platform in Retail Environment. In International Conference on Ubiquitous Computing and Ambient Intelligence (pp. 496-502). Springer, Cham.

The following are other articles in international journals indirectly related to the work done in this PhD Thesis:

- Sánchez, B. B., de Rivera, D. S., & Sánchez-Picot, A. (2016). Building unobtrusive wearable devices: an ergonomic cybernetic glove. *J. Internet Serv. Inf. Secur.*, 6(2), 37-52.
- de Rivera, D. S., & Sánchez-Picot, A. (2015). Dynamic price generation and distribution system using ECA rules and ink labeling. *IT CoNvergence PRActice (INPRA)*, 3(1), 31-41.
- Sánchez-Picot, A., & de Rivera, D. S. Collaboration between services using a B2B Platform in the retail environment.

Table 11.1 shows a summary of the number of publications commented in this section.

Table 11.1: Publications

Type of article	Directly related	Indirectly related	Total
JCR	2	5	7
CORE	1	0	1
International Conference	12	4	16
Other articles	2	3	5

Bibliography

- [mas, 2015] (2015). MASON Social Simulator. <https://cs.gmu.edu/~eclab/projects/mason/>. Accessed: 2019-02-12.
- [omn, 2017] (2017). Omnet++ Network Simulator. <https://omnetpp.org>.
- [mat, 2018] (2018). MATLAB. <https://www.mathworks.com/products/matlab.html>. Accessed: 2019-02-12.
- [net, 2018] (2018). NetSim Network Simulator. <https://www.tetcos.com/>. Accessed: 2019-02-12.
- [ns3, 2018] (2018). ns-3 Network Simulator. <https://www.nsnam.org/>. Accessed: 2019-02-12.
- [rep, 2018] (2018). Repast Social Simulator. <https://repast.github.io/>. Accessed: 2019-02-12.
- [riv, 2018] (2018). Riverbed Modeler. <https://www.riverbed.com/gb/products/steelcentral/steelcentral-riverbed-modeler.html>. Accessed: 2019-02-24.
- [swa, 2018] (2018). Swarm Social Simulator. <https://github.com/swarmsim/swarm>. Accessed: 2019-02-12.
- [err, 2018] (2018). The R Project for Statistical Computing. <https://www.r-project.org/>. Accessed: 2019-02-12.
- [Alcarria et al., 2017] Alcarria, R., de Andrés, D. M., Bordel, B., de Rivera, D. S., Sánchez-Picot, Á., and Robles, T. (2017). A service-oriented monitoring system based on rule evalua-

- tion for home automation. In *2017 IEEE International Conference on Consumer Electronics (ICCE)*, pages 329–330. IEEE.
- [Alcarria et al., 2014] Alcarria, R., Robles, T., Morales, A., and Cedeño, E. (2014). Resolving coordination challenges in distributed mobile service executions. *International Journal of Web and Grid Services*, 10(2-3):168–191.
- [Allan, 2010] Allan, R. J. (2010). *Survey of agent based modelling and simulation tools*. Science & Technology Facilities Council.
- [Blackwell et al., 2009] Blackwell, A. F., Rode, J. A., and Toye, E. F. (2009). How do we program the home? Gender, attention investment, and the psychology of programming at home. *International Journal of Human-Computer Studies*, 67(4):324–341.
- [Blochwitz et al., 2011] Blochwitz, T., Otter, M., Arnold, M., Bausch, C., Elmqvist, H., Jung-hanns, A., Mauß, J., Monteiro, M., Neidhold, T., Neumerkel, D., et al. (2011). 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.
- [Bordel et al., 2019] Bordel, B., Alcarria, R., Sánchez-Picot, Á., and Sánchez-de Rivera, D. (2019). Cyber-physical systems for environment and people monitoring in large facilities: A study case in public health. In *International Conference on Information Technology & Systems*, pages 406–416. Springer.
- [Bordel et al., 2018] Bordel, B., Iturrioz, T., Alcarria, R., and Sanchez-Picot, A. (2018). Provision of next-generation personalized cyber-physical services. In *2018 13th Iberian Conference on Information Systems and Technologies (CISTI)*, pages 1–6. IEEE.
- [Breslau et al., 2000] Breslau, L., Estrin, D., Yu, H., Fall, K., Floyd, S., Heidemann, J., Helmy, A., Huang, P., McCanne, S., Varadhan, K., and Others (2000). Advances in network simulation. *Computer*, (5):59–67.
- [Broman et al., 2015] Broman, D., Greenberg, L., Lee, E. A., Masin, M., Tripakis, S., and Wetter, M. (2015). Requirements for hybrid cosimulation standards. In *Proceedings of the*

- 18th International Conference on Hybrid Systems: Computation and Control*, pages 179–188. ACM.
- [Cao, 2002] Cao, H. (2002). *Development of techniques for general purpose simulators*. PhD thesis, Stanford University Stanford, CA.
- [Cioffi-Revilla, 2014] Cioffi-Revilla, C. (2014). On the quality of a social simulation model: A lifecycle framework. In *Advances in Social Simulation*, pages 13–23. Springer.
- [Climent et al., 2013] Climent, S., Capella, J. V., Blanc, S., Perles, A., and Serrano, J. J. (2013). A Proposal for modeling real hardware, weather and marine conditions for underwater sensor networks. *Sensors*, 13(6):7454–7471.
- [Combs and Others, 2007] Combs, G. and Others (2007). Wireshark. *Web page: <http://www.wireshark.org/>*, pages 2–12.
- [Cook et al., 2009] Cook, D. J., Augusto, J. C., and Jakkula, V. R. (2009). Ambient intelligence: Technologies, applications, and opportunities. *Pervasive and Mobile Computing*, 5(4):277–298.
- [Davidsson, 2002] Davidsson, P. (2002). Agent based social simulation: A computer science view. *Journal of artificial societies and social simulation*, 5(1).
- [Duarte et al., 2006] Duarte, L. M., Kramer, J., and Uchitel, S. (2006). Model extraction using context information. In *International Conference on Model Driven Engineering Languages and Systems*, pages 380–394. Springer.
- [Gyrard et al., 2015] Gyrard, A., Serrano, M., and Atemezeng, G. A. (2015). Semantic web methodologies, best practices and ontology engineering applied to Internet of Things. In *Internet of Things (WF-IoT), 2015 IEEE 2nd World Forum on*, pages 412–417. IEEE.
- [Henderson et al., 2006] Henderson, T. R., Roy, S., Floyd, S., and Riley, G. F. (2006). ns-3 project goals. In *Proceeding from the 2006 workshop on ns-2: the IP network simulator*, page 13. ACM.

- [Issariyakul and Hossain, 2011] Issariyakul, T. and Hossain, E. (2011). *Introduction to network simulator NS2*. Springer Science & Business Media.
- [Jeruchim et al., 2006] Jeruchim, M. C., Balaban, P., and Shanmugan, K. S. (2006). *Simulation of communication systems: modeling, methodology and techniques*. Springer Science & Business Media.
- [Kachan, 2010] Kachan, D. (2010). Integration of ns-3 with matlab/simulink.
- [Kodur and Dwaikat, 2008] Kodur, V. and Dwaikat, M. (2008). A numerical model for predicting the fire resistance of reinforced concrete beams. *Cement and Concrete Composites*, 30(5):431–443.
- [Kuhl et al., 2000] Kuhl, F., Dahmann, J., and Weatherly, R. (2000). *Creating computer simulation systems: an introduction to the high level architecture*. Prentice Hall PTR Upper Saddle River.
- [Law et al., 1991] Law, A. M., Kelton, W. D., and Kelton, W. D. (1991). *Simulation modeling and analysis*, volume 2. McGraw-Hill New York.
- [Lorig et al., 2015] Lorig, F., Dammenhayn, N., Müller, D.-J., and Timm, I. J. (2015). Measuring and comparing scalability of agent-based simulation frameworks. In *German Conference on Multiagent System Technologies*, pages 42–60. Springer.
- [Luke, 2005] Luke, S. (2005). MASON: A Multiagent Simulation Environment. *Simulation*, 81:517–527.
- [Luke et al., 2003] Luke, S., Balan, G. C., Panait, L., Cioffi-Revilla, C., and Paus, S. (2003). MASON: A Java multi-agent simulation library. In *Proceedings of Agent 2003 Conference on Challenges in Social Simulation*, volume 9, page 9.
- [Luke et al., 2004] Luke, S., Cioffi-Revilla, C., Panait, L., and Sullivan, K. (2004). Mason: A new multi-agent simulation toolkit. In *Proceedings of the 2004 swarmfest workshop*, volume 8, pages 316–327. Michigan, USA.

- [Lusch and Vargo, 2014] Lusch, R. F. and Vargo, S. L. (2014). *The service-dominant logic of marketing: Dialog, debate, and directions*. Routledge.
- [Martin et al., 2013] Martin, D., Alcarria, R., Robles, T., and Morales, A. (2013). A systematic approach for service prosumerization in IoT scenarios. In *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2013 Seventh International Conference on*, pages 494–499. IEEE.
- [Martín et al., 2016] Martín, D., Bordel, B., Alcarria, R., Sánchez-Picot, Á., de Rivera, D. S., and Robles, T. (2016). Improving learning tasks for mentally handicapped people using ami environments based on cyber-physical systems. In *International Conference on Ubiquitous Computing and Ambient Intelligence*, pages 166–177. Springer.
- [Martín et al., 2017] Martín, D., Bordel, B., Alcarria, R., Sánchez-Picot, Á., de Rivera, D. S., and Robles, T. (2017). Prosumerization approach to semantic ambient intelligence platforms. In *International Conference on Ubiquitous Computing and Ambient Intelligence*, pages 109–120. Springer.
- [Miller and Myers, 2001] Miller, R. C. and Myers, B. A. (2001). Outlier finding: Focusing user attention on possible errors. In *Proceedings of the 14th annual ACM symposium on User interface software and technology*, pages 81–90. ACM.
- [Morales et al., 2014] Morales, A., Alcarria, R., Martin, D., and Robles, T. (2014). Enhancing Evacuation Plans with a Situation Awareness System Based on End-User Knowledge Provision. *Sensors*, 14(6):11153.
- [Palensky et al., 2014] Palensky, P., Widl, E., and Elsheikh, A. (2014). Simulating cyber-physical energy systems: Challenges, tools and methods. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 44(3):318–326.
- [Pax et al., 2017] Pax, R., Bonett, M. C., Gómez-Sanz, J. J., and Pavón, J. (2017). Virtual development of a presence sensor network using 3d simulations. In *International Conference on Smart Cities*, pages 154–163. Springer.

- [Pax and Pavón, 2015] Pax, R. and Pavón, J. (2015). Multi-agent system simulation of indoor scenarios. In *2015 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pages 1757–1763. IEEE.
- [Pimentel and Reynolds, 2004] Pimentel, R. W. and Reynolds, K. E. (2004). A model for consumer devotion: Affective commitment with proactive sustaining behaviors. *Academy of Marketing Science Review*, 5(1):1–45.
- [Prahalad and Ramaswamy, 2002] Prahalad, C. K. and Ramaswamy, V. (2002). The co-creation connection. *Strategy and Business*, pages 50–61.
- [Quadri et al., 2015] Quadri, I., Bagnato, A., Brosse, E., and Sadovykh, A. (2015). Modeling methodologies for cyber-physical systems: Research field study on inherent and future challenges. *ADA USER*, 36(4):246.
- [Ren et al., 2009] Ren, C., Yang, C., and Jin, S. (2009). Agent-based modeling and simulation on emergency evacuation. In *International Conference on Complex Sciences*, pages 1451–1461. Springer.
- [Ritter et al., 2011] Ritter, F. E., Schoelles, M. J., Quigley, K. S., and Klein, L. C. (2011). Determining the number of simulation runs: Treating simulations as theories by not sampling their behavior. In *Human-in-the-loop simulations*, pages 97–116. Springer.
- [Sánchez et al., 2017] Sánchez, B. B., Alcarria, R., Sánchez-Picot, Á., and Sánchez-de Rivera, D. (2017). A methodology for the design of application-specific cyber-physical social sensing co-simulators. *Sensors*, 17(10):2177.
- [Sánchez-de Rivera et al., 2019] Sánchez-de Rivera, D., Bordel, B., Sánchez-Picot, Á., Martín, D., Alcarria, R., and Robles, T. (2019). Sharing device resources in heterogeneous cps using unique identifiers with multi-site systems environments. In *International Conference on Information Technology & Systems*, pages 153–164. Springer.
- [Sánchez-Picot et al., 2016] Sánchez-Picot, Á., Martín, D., Bordel, B., Alcarria, R., and Sánchez de Rivera, D. (2016). Towards a simulation of AmI environments integrating so-

cial and network simulations. *1st International Workshop on Ambient Intelligence for Large Premises (AmILP)*.

[Sánchez-Picot et al., 2018a] Sánchez-Picot, Á., Martín, D., Bordel, B., and Sánchez-de Rivera, D. (2018a). Ami environments simulations approach integrating social and network aspects: A case study. *Journal of Ambient Intelligence and Smart Environments*, 10(4):303–314.

[Sánchez-Picot et al., 2016] Sánchez-Picot, Á., Martín, D., de Rivera, D. S., Bordel, B., and Robles, T. (2016). Modeling and simulation of interactions among people and devices in ambient intelligence environments. In *2016 30th International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, pages 784–789. IEEE.

[Sánchez-Picot et al., 2018b] Sánchez-Picot, Á., Sánchez-de Rivera, D., Robles, T., and Jiménez, J. (2018b). Time analysis of the integration of simulators for an ami environment. In *International Symposium on Distributed Computing and Artificial Intelligence*, pages 157–164. Springer.

[Sargent, 2013] Sargent, R. G. (2013). Verification and validation of simulation models. *Journal of simulation*, 7(1):12–24.

[Siraj et al., 2012] Siraj, S., Gupta, A., and Badgujar, R. (2012). Network simulation tools survey. *International Journal of Advanced Research in Computer and Communication Engineering*, 1(4):199–206.

[Sokolowski and Banks, 2011] Sokolowski, J. A. and Banks, C. M. (2011). *Principles of modeling and simulation: a multidisciplinary approach*. John Wiley & Sons.

[Stylos et al., 2004] Stylos, J., Myers, B. A., and Faulring, A. (2004). Citrine: providing intelligent copy-and-paste. In *Proceedings of the 17th annual ACM symposium on User interface software and technology*, pages 185–188. ACM.

[Vargo and Lusch, 2004] Vargo, S. L. and Lusch, R. F. (2004). Evolving to a new dominant logic for marketing. *Journal of marketing*, 68(1):1–17.

- [Winsberg, 2018] Winsberg, E. (2018). Computer simulations in science. In Zalta, E. N., editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, summer 2018 edition.
- [Xu et al., 2017] Xu, Y., Luo, X., Wang, W., and Zhao, W. (2017). Efficient dv-hop localization for wireless cyber-physical social sensing system: A correntropy-based neural network learning scheme. *Sensors*, 17(1):135.
- [Zhou et al., 2012] Zhou, P., Zheng, Y., and Li, M. (2012). How long to wait?: predicting bus arrival time with mobile phone based participatory sensing. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*, pages 379–392. ACM.