Faculdade de Engenharia da Universidade do Porto

An Integrated Framework for Multi-Paradigm Traffic Simulation

José Luís Pereira Macedo

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Rosaldo J. F. Rossetti, PhD (Assistant Professor)

Second Supervisor: Zafeiris Kokkinogenis

February 25, 2013

An Integrated Framework for Multi-Paradigm Traffic Simulation

José Luís Pereira Macedo

Mestrado Integrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

Chair: Doctor António Augusto de Sousa

External Examiner: Doctor Paulo Jorge Pinto Leitão

Supervisor: Zafeiris Kokkinogenis

February 25, 2013

Abstract

The increase of traffic and transport demand witnessed in the last decades is intimately connected with the main problems that we face today. Traffic congestion affects not only the economic activity of cities but is also responsible for air quality and global warming problems. In fact, vehicles emissions are one of the major causes of the green house effect.

In this sense, incentives and investments on public transports as well as research on more eco-sustainable solutions, have been performed on attempt to minimize both the air pollution and congestions problems. One of the approaches currently investigated and implemented to provide an eco-sustainable solution is related to the employment of electric buses powertrain in metropolitan transportation as an alternative to internal combustion engine buses.

However, there are still open issues related to the consumption of energy and other performance measures for considering the adoption of electric buses in urban scenarios as a cost-effective solution. An important aspect in evaluating the performance and adequateness of such vehicles is the fact of being immersed into an urban environment context. That is, a route having many positive elevations or a traffic congestion situation will affect directly the autonomy and performance of the vehicle. Albeit there are different tools and models to assess the behaviour of electric buses, such evaluations often lack the aforementioned integration with the traffic dynamics.

This work presents a distributed architecture for electric bus powertrain simulation within a realistic urban mobility context. Such a platform wants to offer a valid tool to traffic managers and practitioners for analysing how traffic flow and its dynamics affect the performance of the electric bus when there are obstructions or intense traffic conditions. The proposed simulation framework can be multi-faceted. As a matter of fact it can be used, not only as electric vehicle evaluation tool, but also as a planner for charging point distribution.

For the implementation of the integrated platform the SUMO (Simulation of Urban MObility) microscopic traffic simulator has been coupled with a model of an electric bus powertrain designed in MatLab/Simulink environment. SUMO is an open source simulator with multi-modal traffic feature capabilities that allows the simulation of various types of vehicles.

The integration follows one first approach with the adaptation of the TraSMAPI (Traffic Simulation Management Application Programming Interface) framework to comprise two different simulators at once.Then it is followed the (HLA) High Level Architecture approach for distributed simulation. The electric bus engine and both integration approaches has been validated using field test experimental data. Both the electric bus engine and the integration has been validated using field test experimental data.

Resumo

O aumento do fluxo de tráfego verificado nos últimos anos está diretamente relacionado com os principais problemas com que nos deparamos nos dias de hoje. O congestionamento de tráfego afecta não só a atividade económica das cidades, como é também responsável por problemas relacionados com a qualidade do ar e com o aquecimento global. De fato, as emissões provenientes dos veículos são uma das maiores causas do efeito estufa. Com o objetivo de minimizar a poluição atmosférica e o congestionamento urbano, têm sido realizados investimentos que passam por investigação e incentivos, na pesquisa de transportes públicos que sejam mais económicos, eficientes e sustentáveis. Uma das aplicações actualmente existente para fornecer uma solução eco-sustentável está relacionada com inserção de autocarros eléctricos nas redes rodoviárias dos grandes centros metropolitanos, em alternativa aos autocarros de motor de combustão actualmente utilizados.

No entanto, ainda existem questões relacionadas com o consumo de energia e outras medidas de desempenho, o que coloca em causa a adoção do autocarro elétrico como sendo uma solução rentável. Um aspeto importante na avaliação do desempenho destes veículos é o fato de estarem, ou não, inseridos num ambiente de tráfego urbano. Certamente, uma topologia bastante acentuada vai influenciar directamente a autonomia e a performance do autocarro. Contudo, apesar de existirem diversas ferramentas que modelam o comportamento de um autocarro eléctrico, as análises por si efetuadas não contemplam algumas das características intrínsecas do ambiente de tráfego urbano.

Neste trabalho é apresentada uma arquitetura distribuída para a simulação de um autocarro elétrico num contexto urbano de transporte realista. Esta ferramenta é importante para a análise da influência do tráfego urbano no desempenho do autocarro elétrico em situações de acidente ou de tráfego intenso.

A plataforma de simulação proposta é multi-facetada, já que pode ser usada não só como uma ferramenta de avaliação de autocarros elétricos, mas também como um ferramenta de planeamento para a distribuição de postos de carregamento. Para a implementação desta aplicação integrada, foi utilizado o simulador microscópico de tráfego SUMO (Simulation of Urban Mobility), um simulador de código aberto que suporta a simulação de vários tipos de veículos, ao qual foi anexado um modelo de um autocarro elétrico implementado no ambiente modular do MatLab/Simulink.

A integração cumpre os requisitos definidos pelo conceito da High Level Architecture (HLA) para simulação distribuída. Tanto o modelo do autocarro elétrico como a ferramenta de integração foram validados usando dados de teste recolhidos num ambiente real.

Acknowledgements

First of all, I would like to thank to my family and friends for their encouragement and support that contributed to this important achievement which is the conclusion of a Master's Dissertation.

I would like to thank to my supervisor Professor Rosaldo Rossetti, who has believed in my potential from the first day, for his professionalism, advices and friendship.

Many thanks to my co-supervisor Zafeiris Kokkinogenis, for his assistance and companionship all over the period of preparation of this work, and for his wonderful tiramisu that helped sweeten up difficult and stressed moments.

I would like to thank to the Faculty of Engineering - University of Porto (FEUP) and the Artificial Intelligence and Computer Science Laboratory (LIACC), for the reception, support and encouragement given; In particular to Professor Eugénio Oliveira, the heart of the laboratory, and Professor Augusto Sousa, the director of the course, for the opportunity to accomplish this Master's dissertation.

Last but not least, the most special thanks go to my girlfriend, who have motivated me over the last years with her kindness and charm. Thank you for all your concerns and advices that have helped me to end up with this Dissertation.

José Macedo

*“No one wants to learn by mistakes,
but we cannot learn enough from successes to go beyond the state of the art.”*

Henry Petroski

Contents

List of Figures

LIST OF FIGURES

List of Tables

Abbreviations

API	Application Programming Interface
EBPS	Electric Bus Powertrain Subsystem
FOM	Federation Object Model
GUI	Graphical User Interface
HLA	High Level Architecture
IEEE	Institute of Electrical and Electronics Engineers
ITS	Intelligent Transportation Systems
MATLAB	MATrix LABoratory
M&S	Modelling and Simulation
OMT	Object Model Template
RTI	Run-Time Infrastructure
SOM	Simulation Object Model
SUMO	Simulation of Urban MObility
TraCI	Traffic Control Interface
TraSMAPI	Traffic Simulation Management Application Programming Interface
XML	Extensible Markup Language

Chapter 1

Introduction

In the last decades it has been witnessed a large increase in traffic and transport demand that has created and aggravated capacity problems in the infrastructure causing traffic congestions and delays. Problems in the traffic system have a large impact on almost all areas of economic activity since the flow of people and goods between cities is directly related to the road network [ARS07].

Furthermore, traffic congestion affects not only the welfare of the citizens from the economic point of view but is also related to their health status both psychological, due to stress accumulated during their travels, and physical due to high air pollution levels. In fact, a problem associated with the increasing use of personal vehicles is the emissions. According to the 2009 Urban Mobility Report [SL09] the congestion led urban Americans to travel 4.2 billion hours more, which resulted on 2.8 billion gallons of extra fuel, an increase of more than 50% over the previous decade.

The green house effect, also known as global warming, is a serious issue that we have to face. There has been increased tensions in part of the world due to the energy crisis. Government agencies and organizations try to develop more stringent standards for the fuel consumption and gas emissions through reduction of the congestion on network infrastructures [Sov10]. However, this is no longer regarded as a problem confined only to large metropolitan areas. Currently, the traffic problems typical to densely developed urban areas began to spread to the suburbs as the people move away from the traditional centre city work pattern [Dow04]. In addition, advances in automotive technology have allowed more people to drive which has led to traffic congestion even in small towns.

There is, of course, a diversity of different solutions trying to tackle congestion problems. As congestion begins to occur when the amount of traffic within a road network is approaching its maximum capacity, the most obvious solution is to increase the network capacity [ZY04]. This can be done in several ways, such as building new roads, extending the existing ones and adjusting the speed limit of roads to increase their capacity. However, creating new roads or adding additional capacity to the existing ones can be expensive, time consuming, can cause environmental and social impacts and is not guaranteed that it solves the problem, as demonstrated by Braess' paradox [Bra68, Mur70]. Moreover, road traffic is growing faster than capacity and is expected to continue to do so if no measures are imposed to limit traffic growth.

Various policy-based methods to reduce road traffic have been attempted, like, for instance, introduced a congestion charge, a system which works by charging motorists who travel into and out of a designated area within certain hours. But these options are not always entirely viable; numerous criticisms including an increased risk of crashes due to speed differences and violation of principles of equality have been levied against this idea [Ye12]. Another attempted solution used for various countries was through the layout of road-networks. The introduction of one-way systems and ring-roads, and the use of road systems that curve and merge into each other rather than perpendicular junctions (roundabouts are a good illustration) are some examples of these attempted [Ty10].

However, the increase in traffic volumes combined with often short distances between intersections requires the adoption of a systems analysis approach to properly address traffic congestion. Often traffic congestion is not the result of excessive traffic, but the result of overlapping bottleneck locations. The spillover effect of traffic congestion from one location to another makes inefficient conventional engineering methods [MM01].

In this sense, incentives and investments on public transports as well as research on more eco-sustainable solutions have been performed as attempts to minimize both the air pollution and congestion problems. One of the approaches currently investigated and implemented to provide an eco-sustainable solution for the public transport is related to the employment of electric bus powertrains in metropolitan transportation as an alternative to internal combustion engine buses [UBW^{+}10]. However, there are still open issues related to the consumption of energy and other performance measures for considering the adoption of electric buses in urban scenarios as a cost-effective solution [Mac].

An important aspect in evaluating the performance and adequateness of such vehicles is the fact they are to be in immersed into an urban environment context. That is, a route having many positive elevations or a traffic congestion situation will directly affect the autonomy and performance of the vehicle.

Alongside ITS (Intelligent Transportation Systems), other concept that has been gaining a great importance in traffic and transportation domain is the simulation concept. The use of computer simulations has proved to be a crucial assistance to traditional traffic engineering analysis methods in fully understanding the dynamics of traffic movement and control processes [WS06]. In fact, it allows the prediction of the impact of new solutions before being applied in real scenarios and also enables the execution of experiments which may be impossible, either due to their excessive cost or consequences [Pur99]. For example, traffic simulation can be used to estimate the impacts at network level of ITS candidate solutions. In this way, one can easily experiment with penetration rates or system settings to create hypothetical future scenarios.

Traffic simulation uses different computational models to represent different domain abstractions. Each of these models characterizes a level of granularity of the real system, depending on the perspective and the type of analysis one intends to perform [IL02]. These models serve for different purposes, and each of them has its own advantages and disadvantages over the others. For example, microscopic models which provide a detailed representation of the traffic process can simulate to the granularity of a single vehicle but cannot simulate efficiently large-scale traffic networks.

In complex networks, requiring a large amount of input data, the use of a microscopic model results on a tremendous computational cost. In contrast a macroscopic model captures traffic dynamics in lower detail being most suitable for modelling large networks, but failing to capture the behaviour of vehicles at junction level e.g. traffic signal control of an intersection [MSLZ11]. Thus, no model can be completely replaced by another one as each of it bears information for a specific level of resolution.

Traditionally, the various types of simulation tools such as traffic simulators use a specific type of model and, moreover, are used as standalone tools. This leads experts to work separately on different tools and models when the problem is often complex and therefore requires an integrated analysis as well. There are some traffic simulators capable of combining different simulation models (e.g. AIMSUN [Tra] , TransModeler [Cal]). However, most of them are commercial and mainly focus on macroscopic/microscopic integration as they claim. None of them embodies a nanoscopic aspect of the system.

The option of creating a new simulator, or extending an open-source one in order to integrate different types of traffic simulation models could not be a good approach. On one hand, such approach requires an enormous and complex work that would need a very detailed validation without quarantining flexibility and interoperability. On the other hand, there are many simulators each one designed for a specific simulation model which have been used in a great amount of studies and so they are already validated and well accepted.

An interesting way to work around the lack of this kind of integration would be to get simulators that implement traffic models of different resolution working together allowing data exchange and thus allowing different types of analyses. In fact, there are some interesting works using combinations of different simulators for integrating different types of models [YLBO07, CPD$^+$00, DRE02, CH09]. However, none of the applications developed to date are sufficiently generic to being able to easily add new simulation tools without the risk of losing consistency.

1.1 Motivation and Objectives

In order to analyse the adoption of electric buses as a cost-effective solution, it is needed an evaluation of the performance and adequateness of such vehicles while immersed into an urban environment context. Albeit there are different tools and models to assess the behaviour of electric buses, such evaluations often lack the aforementioned integration with the traffic dynamics [Mac]. This work will present a distributed architecture for electric bus powertrain simulation within a realistic urban mobility context. Such a platform will be important for analysing how traffic flow and its dynamics affect the performance of the electric bus when there are obstructions or intense traffic conditions.

Thus, the main objective of this thesis is to study the possibility of integrating microscopic and nanoscopic traffic simulation models and evaluate the advantages that can be achieved on integrated studies. More specifically, it is an intention of this work the integration of two different simulators: the Matlab/Simulink model of an electric bus subsystem (for simulating the consumptions of an electric vehicle) and SUMO [BBEK11] a microscopic traffic simulator (for simulate the urban traffic conditions under which the electric vehicle should be evaluated).

It is also an aim of this work to study the concepts and the potential of using High Level Architecture (HLA) to interconnect different simulation systems. For last, it will be analysed the influence of drivers' behaviour in the electric engine consumptions.

1.2 Thesis structure

The report is structured as follows:

In this chapter, an introduction to the subject of work and most important goals to achieve are presented.

Chapter 2, will cover some background to the subject of modelling and simulation as well as distributed simulation where the HLA concepts are introduced. Furthermore it will present a deeper overview on traffic simulation and its different simulation models, and some related works related with the integration of the different types of these models.

Chapter 3, starts with a recall of the problem definition and objectives of the project. After this, is presented the architectural solution, within HLA concepts, along with its fundamental issues to be addressed. Also, a methodological approach is established;

Chapter 4 introduces the software package used on the development of the proposed solution.

In Chapter 5 is presented the main implementation steps of the solution for the encountered problems and aforementioned requirements.

In Chapter 6 some functionality and performance tests to the integration implementation are performed, and its results are discussed. Furthermore, a test-bed is presented to highlight the potential of the developed framework for integrated studies.

Chapter 7 concludes the document depicting the main contributions, final remarks and future work.

Chapter 2

Literature Review

This chapter will explore the background concepts needed for a better understanding of the project and its position respect to other similar works in the field. It starts presenting some concepts about simulation in general before moving to a more specific scope such as traffic domain simulation. Afterwards, related work on the integration of different models in traffic domain is presented. Finally, background concepts of distributed simulation using the High Level Architecture approach and its application in civil domain are presented.

2.1 Modelling and Simulation Overview

First of all, it is important to introduce the concept of system in the M&S context. According to Schmidt and Taylor [ST70], a system is a *collection of entities*, e.g. people or machines that act and interact together toward the accomplishment of some logical end. In practice, the components of a system depend on the scope and objectives of a particular study.

The "definition" of the collection of entities in a system is a matter of perspective. That is, such collection that composes the system in a given study might be only a subset of the overall system for another one. For example, if one wants to study the number of employees that a restaurant needs, to provide an adequate service to its customers, the system can be defined by the employees and the customers that are being served. On the other hand, if one wants to include, the logistics of the restaurant, the necessary entities have to be added to the system [LK91].

A first overview to the whole concept of Modelling & Simulation (M&S) is provided in order to start justifying the motivation behind this work.

In the literature there is not a unique definition for M&S. Rather, since the very first definition, each domain area and scientific field, within or related to the M&S discipline, tends to define with its own perspective the general framework that covers this discipline. The Merriam-Webster On-Line Dictionary defines simulation as "the imitative representation of the functioning of one system or process by the functioning of another". Maybe, in a more methodological way, one could say that simulation is the imitation of some real entity, object, state of output, or process over time that represents certain features or behaviours of the selected physical or abstract system. This means that to determine how an actual system function and perform, we would build a model of the system and observe how the model operates [Mar97].

Since decades, the development, analysis and experimentation with models are the basic tools of science and applied systems in economy and industry. Modelling is essentially the development of a model as a representative of a system, or better, the process of producing a model of it. A model is a representation of the structure and operation of some system of interest. It is similar to, but simpler than the system it represents [Rob07]. One purpose of a model's usage is for helping the modeller to predict the effects that changes in the system can provoke. On the one hand, a model should be a close approximation to the real system and incorporate most of its relevant features. On the other hand, it should not be so complex that makes it impossible to understand and experiment with it. A good model is a judicious trade-off between realism and simplicity [Sar05]. An important issue in modelling is model validity. Model validation techniques include simulating the model under known input conditions and comparing model output with system output.

A simulation of a system is mainly the execution of a model of the system, but not only. As a matter of fact simulation is rather an engineering process in which the model operation is one of the steps. A well-specified working process that guides model development and usage should define which steps have to be performed, which documents and results have to be delivered in what phase of the study. Having a definitive approach for conducting a simulation study is critical to the study success in general and to developing a valid model in particular [Car04]. The model can be reconfigured and experimented with; usually when it is impossible, expensive or impractical to do in the system it represents. The operation of the model can be analysed, and thus, properties concerning the behaviour of the actual system or its subsystem can be inferred. In its broadest sense, simulation is a tool to evaluate the performance of a system, existing or proposed, under different configurations of interest and over long periods of time.

Thus, simulation is used before an existing system is altered or a new system built, to reduce the chances of failure to meet specifications, to eliminate unpredicted tie-up, to prevent under or over-utilization of resources, and to optimize system performance. For instance, simulation can be used as what-if scenarios generator to answer questions like: What is the best design for a new transportation network? What are the associated resource/costs requirements? How will a transportation network perform when the traffic load increases by 50%? How will a new traffic control algorithm affect its performance? What will be the impact of link congestion over the performance of an electric motor operation [ZPK00]?

According to practitioners and researchers, simulation modelling and analysis is one of the most frequently used operations research techniques. When correctly used, simulation modelling and analysis makes it possible to:

- Obtain a better understanding of the system by developing a mathematical model of a system of interest, and observing the system's operation in detail over long periods of time.

- Test hypotheses about the system's feasibility.
- Compress time to observe certain phenomena over long periods or expand time to observe a complex phenomenon in detail.
- Study the effects of certain informational, organizational, environmental and policy changes on the operation of a system by altering the system's model; this can be done without disrupting the real system and significantly reduces the risk of experimenting with the real system.
- Experiment with new or unknown situations about which only weak information is available.
- Identify the "driving" variables - ones that performance measures are most sensitive to - and the interrelationships among them.
- Identify bottlenecks in the flow of entities (material, people, etc.) or information.
- Use multiple performance metrics for analysing system configurations.
- Employ a systems approach to problem solving.
- Develop well-designed and robust systems and reduce system development time.

Applications of simulation abound in the areas of administration, military, computer and communication systems, manufacturing, transportation, health care, ecology and environment, sociological and behavioural studies, biosciences, epidemiology, services, economics and business analysis. Each of these application areas can perceive simulation under different perspective. The following Table 2.1 presents the perspectives under of which M&S can be perceived: purpose of use, problem to be solved, connectivity of operations, types of knowledge processing, and philosophy of science [Ö09].

Perception with respect to	**Perceptions of simulation**
Purpose of use	Perform experiments for: Decision support, Understanding, Education, Training, Entertainment
Problem to be solved	Black box perception (M&S is an infrastructure to support real-world activities)
Connectivity of operations	Standalone simulation; Integrated simulation (symbiotic simulation)
Types of knowledge processing	Computational activity; Systemic activity; Model-based activity; Knowledge generation activity; Knowledge processing activity
Philosophy of science	Simulation supports and enriches modern scientific thinking

Table 2.1: Perception of M&S from different perspectives adapted from [Ö09]

A model construction can be intended to solve a specific problem within a domain. Thus, there may be a number of different models for the same domain, each model complying with the characteristics of a particular problem. A. Law and D. Kelton [LK91] divide simulation models in three classes:

- **Static vs. Dynamic Models -** Static models are either models of time-independent systems or models of a system at a particular time. Dynamic models are those that represent a system as it evolves over time.

- **Deterministic vs. Stochastic Models -** Stochastic means random, determined by chance. Models that rely on the generation of random variables in deciding how to change state are stochastic. Every time such a model is executed a different result is yield. If the execution of the model continues for many times it will give a measure of variability in the process as predicted by the model. With a deterministic one the assumptions and equations selected define the results. A deterministic model is one in which every set of variable states is exclusively determined by parameters in the model and by sets of previous states of these variables. Consequently, deterministic models perform the same way for a given set of initial conditions. Deterministic models can describe behaviours on the basis of some physical law. For example, the planets move around the sun according to Newton's laws and their position can be predicted with great accuracy into the future [Nel].

- **Continuous vs. Discrete Models -** Discrete models are ones in which the state variables change instantaneously at separated points in time while continuous models are ones in which the state variables change continuously over time. Some authors [ZPK00] make distinction between discrete-event models and discrete-time models. The later are a sub category of the discrete-event models in which all time steps are considered for all elements of the model. Discrete-event models that are not discrete-time models consider only those time steps at which state changes occur. An example of a discrete system is the supermarket system, since state variables, such as the number of clients at the supermarket, only change when he arrives or departs. The movement of a car within a city is an example of a continuous system since state variables such as position and velocity can change continuously over time.

Some set of relationships that compose a model are simple enough to be possible to use mathematical methods such as algebra or probability theory, to obtain the exact information. However, most real-world systems are too complex to allow realistic models to be evaluated by analytic methods. For these complex systems, the simulation is performed using computational means, in order to evaluate the model numerically and estimate realistic model characteristics. Figure 2.1 illustrates the different ways of studying a system.

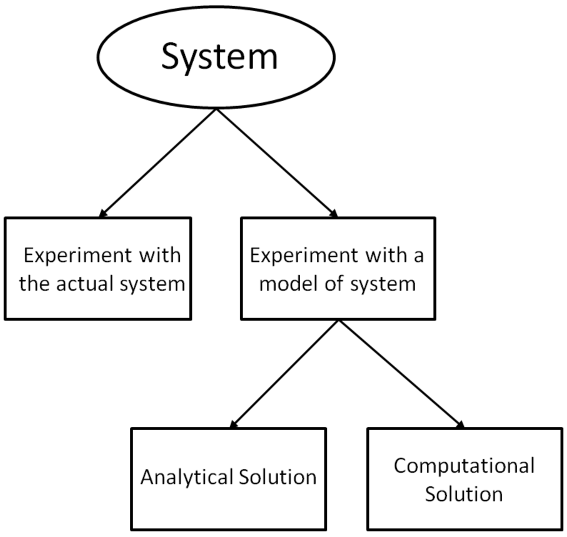

System
Experiment with the actual system
Experiment with a model of system
Analytical Solution
Computational Solution

Figure 2.1: Ways of systems analysis

2.2 Distributed Simulation

2.2.1 Overview

With the rapid advances being made in computer and software, several new branches appeared in the computer simulation domain. One of these branches is distributed simulation. Distributed simulation refers to technologies "that enable a simulation program to execute on a computing system containing multiple processors, such as personal computers, interconnected by a communication network" [Fuj01]. The goal for distributed simulation is to provide and facilitate interoperability and reusability of heterogeneous simulation systems. This objective is supported by the arrival of the High Level Architecture concept. HLA provides for the first time a real industry standard which aims interoperability for a wide range of simulation systems and applications.

2.2.2 HLA Concepts

The HLA is an IEEE (Institute of Electrical and Electronics Engineers) standard software developed to provide a common architecture for distributed modelling and simulation (M&S). It is a component-based software architecture that addresses the interoperability and reusability of different models and units of simulations, and offers time management interoperability as well [KDW00]. In order to facilitate interoperability and reusability, HLA differentiates between the simulation functionality provided by the members of the distributed simulation and a set of basic services for data exchange, communication and synchronization.

Architecture and Components

In HLA, every participating application is called federate, and these entities can interact with each other within a federation. A federation can be seen as a set of federates acting together in a distributed simulation to achieve a certain objective.There are three main components that comprise HLA:

- **Federate Interface Specification**
- **Framework and Rules**
- **Object Model Template Specification**

The HLA Framework and Rules is the set of rules that must be obeyed to ensure the proper interaction of federates within a federation. These rules must be unchanged across all the simulation units as they define the overall architecture. They also define the responsibilities of federates and federation. There are five rules for federates and other five to federations.The definition and description of each rule is available in [IEE10b].

The HLA Federate Interface Specification describes the services which federates have to use for communicating with others. This communication is always made through a middle-ware structure, known as Run-Time Infrastructure, which provides the essential building ground for the software developers. The interface specification describes which services a federate can use and which services it has to provide [IEE10a]. In order to establish the interaction between federates and the Run-Time Infrastructure (RTI), the concept of ambassador is used. Ambassadors are objects that have the methods needed by the participants for performing communication. So, federates communicate with the RTI using its ambassador as an interface. Figure 2.2 illustrate the described concepts.

The HLA Object Model Template Specification describes the format and syntax of the data transferred between federates. This data exchange is represented in the form of object class and the two types of object exchange are Object Class and Interaction Class. The first one contains the shared information within federation that persists during the run time. The second one, contains the sent and received information between federates. This component defines the object template data that all simulation unit needs to use in order to exchange data with each other [IEE10c].

2.3 Simulation in Traffic and Transportation Domain

One of the major problems facing transportation engineers and urban planners is that of predicting the impact of given transportation scenarios. For decades, the use of simulation methodologies in Transportation Systems field is widely acclaimed. If one looks for current transportation state in urban scenarios, high traffic saturation levels due to the increasing demand and a not-optimized transportation planning is evident [YCC10]. Thus, computer models are widely used in traffic and transportation system analysis, with a variety of applications from scientific research to planning, training and demonstration.

Traffic simulation tools aim not only to deal with undesired events as mentioned above, but also to generate scenarios, optimize control, and predict network behaviour at the operational

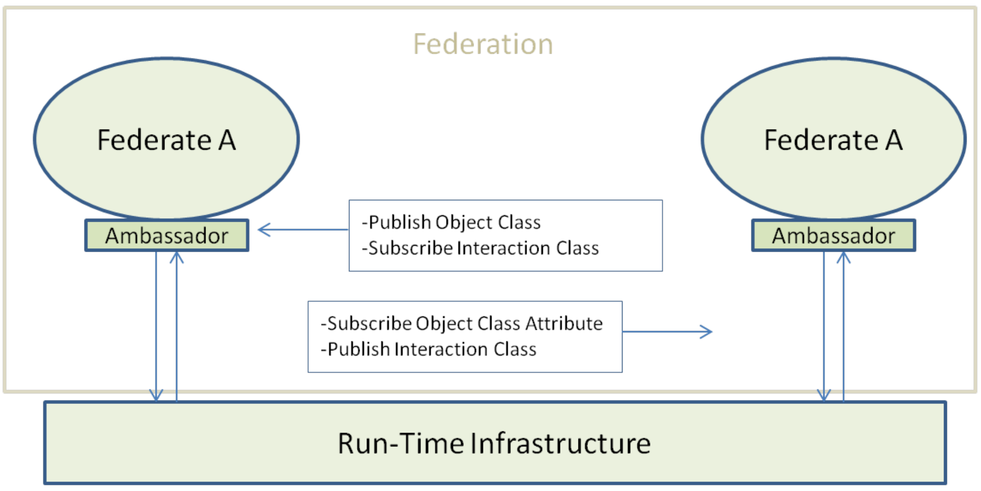


Figure 2.2: HLA's Functional Architecture

level. However, for the same domain there are a great variety of models, each one representing a different abstraction of it.

In fact, each person has its own way of seeing things, its own point of view over a particular scenario. Thus, each analyst has his own perspective over a specific problem and needs to see through that point of view to be capable of analyse that problem. So, each simulation model is an abstraction of the domain that tries to represent a specific perspective for any specific kind of experts.

After some years of research in traffic flow and the application of its findings to the planning and management of traffic, the discipline has developed a wide variety of methods and tools it can use. For an overview of the state of the art in traffic flow research, see [May90], [Dag97] or [GRM97]. There exist a large number of models, and they are usually characterised by the level of detail in which they describe the traffic processes. As said before, models are classified into different categories depending on the level of detail that represents. Macroscopic, mesoscopic, microscopic and nanoscopic are these categories. Figure 2.1 illustrate the different granularities.

2.3.1 Macroscopic Models

Macroscopic models describe traffic at a high level of aggregation such as flows or densities. These flows are the number of vehicles that pass through a certain road per hour. However these kinds of models do not consider the constituent parts of that flow such as the vehicles. Macroscopic models such as the LWR model [LW55] use differential equations to formulate relationships among traffic flow density. These equations describe traffic flow density like flows in fluids or gases, and therefore, its solution can be obtained through simulation.

In short, macroscopic models only deal with the flow of traffic which made them good for a large and complexes networks analyses. It is useful for route planning and has a lower computational cost compared with the other models, but fails to capture the individual behaviours and

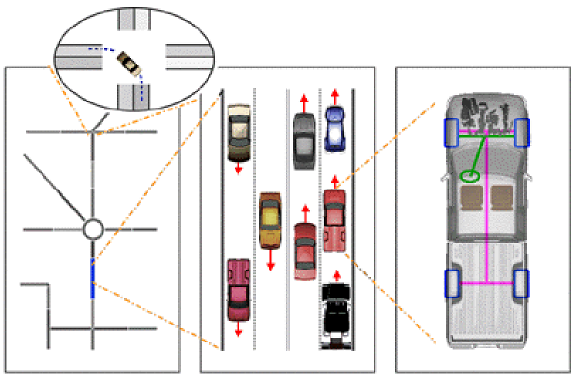


Figure 2.3: The different simulation granularities; from left to right: macroscopic, microscopic, nanoscopic (within the circle: mesoscopic) [KHWR02]

detailed situations of traffic [Bur04].

2.3.2 Mesoscopic Models

Mesoscopic models fill the gap between macro and micro models, they normally describe traffic entities at a high level of detail, but their behaviour and interaction are in a lower level of detail. In mesoscopic model, vehicles can be grouped in packets, which are routed through the network and are treated as one entity. Other paradigm is that of individual vehicles that are grouped into cells to control their behaviour. The cells traverse the link and vehicles can enter and leave cells when needed, but not overtake [Bur04].

2.3.3 Microscopic Models

Microscopic models have a more detailed representation of the traffic than macroscopic ones. These models describe the behaviour of the entities that make up the traffic stream as well as their interactions. In microscopic models, the level of detail goes till the individual behaviour of vehicles, their interaction with each other and with the road network. For that, these models are capable of perceiving some rules of the vehicles' behaviour such as when a vehicle accelerates, decelerates, changes its lane and chooses or changes their routes to their destinations. Among other some well-known model in the literature are the car-following model [OT04], lane-change model [BACT06], and the route-choice model [Pra09] are the main methods used to determine that vehicle's behaviour.

These types of models are widely used in analyses of detailed traffic situations such as traffic lights control. However, using these types of models on large and complex roads networks could be an impossible task due to the high computational cost that it requires [Bur04].

2.3.4 Nanoscopic Models

A new trend of traffic simulation is the nanoscopic model which extends the capabilities of three basic components of microscopic simulation: vehicle modelling; vehicle movement modelling; and driver behaviour modelling [DP08].

It is mostly used in autonomous driving and a strictly relationship with automated robotic, because needs to simulate sensors and vehicles constitutes parts. Controls and great improves already has been done on this field. In this paper [FRBR09] is observed great potential in using robotic simulators on autonomous driving field, motivating an information exchange among robotic and traffic study groups.

2.3.5 Distributed Simulation and integrated models in Traffic and Transportation Domain

As has been said before, distributed simulation allows dividing computational efforts to improve simulation performance. Over the past few years, distributed simulation in traffic domain has been widely used in studies of most varied fields.

In fact, with the emergence of the Intelligent Transportation Systems (ITS) more and more studies needs to be made with a great level of detail, leading planners and analysts to use microscopic models rather than macroscopic ones. However, microscopic simulations results in a greater computational effort, being almost impossible to be performed in large urban networks without distributing that effort.

In the literature there are two main focus of distribution in simulating the traffic and transportation domain. The first, is when one want to distribute mechanisms, with decision making capabilities, within simulation, (e.g. traffic light agents or driver agents) [WUW$^+$12, FCD12, VO11].

The second one applies when one want to distribute the models among different simulation [LMJR04]. The integration of simulation models, viewed as the coordination and data exchange between different simulators, is a problem to which some approaches have been suggested with diverse motivations and application domains. The main difficulties presented on literature are the data conversion between the different simulation models due to the different paradigm they represent.

As it has been repeatedly mentioned, the necessity of having an integrated tool to represent a system rises from the need for multifaceted analysis of it.

In [SC05] authors consider the integration of macro and microscopic models for the analysis of urban transportation systems. Their concern is based on the advantages of using a macroscopic model for the representation of large-scale networks, and, contemporary, the high resolution in details that can be yield by a microscopic model. They have applied this approach in the analysis of sub-areas that are part of a large macroscopic network, in order to represent detailed design changes as well as traffic management schemes that cannot be treated explicitly by the macroscopic model. In [SWL11] authors present a real-time algorithm for modelling large-scale traffic using both continuum (macroscopic model) and agent-based (microscopic model) methods. It is presented some techniques for dynamic coupling of discrete vehicle simulation with the vehicle aggregated behaviour of continuum model. Figure 2.4 shows the interface of the integration framework of these models. However, the focus of this work is in terms of visualization performance rather as analysis tool. On this purpose [MCBB98] present a macro-microscopic approach for the evaluation of the traffic assignment models in transportation planning. [ZDHB09] describe a traffic control framework for emissions by integrating macroscopic traffic flow models with a microscopic emission and fuel consumption model.

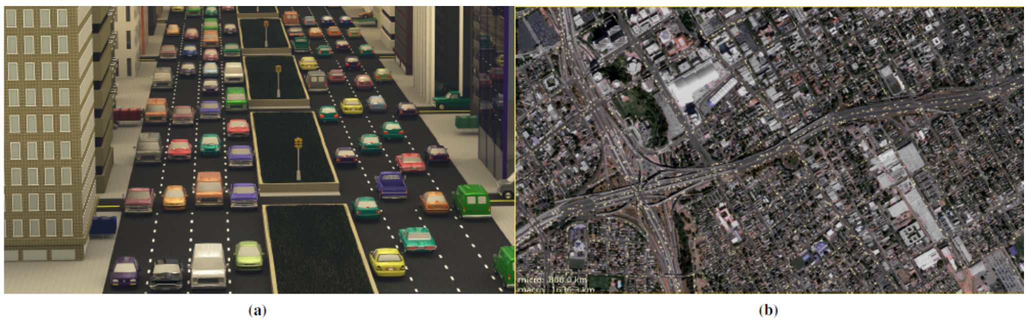


Figure 2.4: (a) Interactive 3D visualization of urban traffic; (b) Augmenting a satellite earth map of a metropolitan region with real-time moving traffic consisting of tens of thousands of vehicles using our method [SWL11].

Also in [MSLZ11] a simple transformation methods are analysed to translate the variables parameters between both models. Furthermore, requirements and design principles for specifying and realizing multi-resolution, are introduced in [YLBO07]. Here, a multi-model specification formalism based on graph of models is suggested along with design precepts to enable flexible dynamic model updating. The notion of multi-simulation is also introduced to enable exploratory simulation using various types of multi-models.

Concerning the integration of a microscopic and mesoscopic models, Burghout [Bur04] presents a mesoscopic traffic simulation model, particularly suited for the development of integrated meso-micro traffic simulation models. More related work to integration of this two simulation models are presented by the Shi and Ziliaskopoulos [SZ06] as well as Yang and Morgan [YM06].

With the advent of the vehicular networks, transportation community had to face the requirement of integrating ad-hoc wireless communication models to the traffic microscopic simulators in order to emulate the network infrastructure for testing the new-type ITS solutions. Among the proposed solutions in the literature are the iTetris framework [GTL$^+$09] and the Veins project [SD08]. Both of the frameworks use SUMO [BBEK11] as traffic simulator.

Another good work concerning microscopic and nanoscopic models, is the one proposed by Martin Adelantado et al [AOC]. It is shown the combining of X-Plane flight simulator, Google Earth browser and the High Level Architecture for evaluating environmental impact of innovative air transport concepts around airports. In [KSSM98] is presented a distributed HLA-based traffic simulation of a nanoscopic model of driver's behaviour and a microscopic simulator. The authors provide a proof-of-concept prototype of a driver simulator with a microscopic traffic simulator and a visualization module. An integrated framework that aims coupling robotics and a traffic simulator is presented in [PR12]. This work developed an integrated framework enabling autonomous vehicles to be deployed in a rather realistic traffic flow at the same time it simulates all its sensors and actuators. To do so, some modifications were performed on both traffic simulation and 3D simulation environment for robot sensors simulation. The application is distributed, as one computer is used to simulating a large traffic environment whereas other computers simulate single autonomous vehicles that integrates the simulation. In [MSAN11] the model of an electric vehicle engine is embedded into the vehicle model of a microscopic simulator. This study is, to the best of our knowledge, the first attempt to integrate electric vehicle model into a traffic simulation. Although it is a forerunner application, the integration was achieved by performing modifications on the core of the simulators. Thus, this integration is not flexible, and do not allow easy coupling of new simulators or other simulation tools. Also, it doesn't account for the powertrain as a whole.

2.4 Summary

In this chapter was presented some background concepts that are essential to a better understanding of the motivation and context of this thesis. It was emphasized the importance of modelling and simulation in studies of many different domains, allowing analysis that are impossible to perform in real-world systems. It was also presented the concept of distributed simulation and its historical evolution, which is important to understand the emergence of the High Level Architecture.

Furthermore, the literature review has shown that combination of different models of simulation play an important role in traffic and transportation domain. It allows planners and analysts to address complex problems while combining the advantages of each simulation model. However, the existing proposals of integration are often application specific. Given this, the interoperability and reusability notions provided by HLA standards are becoming widely studied in civil applications and proved to be an interesting approach for urban traffic domain.

Chapter 3

Methodological Approach

3.1 Problem Statement

The necessity for greener public transport has found in the electric bus powertrain a potential solution. However, such vehicles are still far from being cost-effective. For example, as in all of the actual electric vehicle solution, the electric bus powertrain's driving range is still low respect to the traditional internal combustion vehicle's autonomy. There are still open issues related to the consumption of energy and other performance measures for considering the adoption of electric buses in urban scenarios as a gainful solution. An important aspect in evaluating the performance and adequateness of such vehicles is the fact of being immersed into an urban environment context. That is, a route having many positive elevations or a traffic congestion situation will directly affect the autonomy and performance of the vehicle.

This thesis's main objective is to study the viability of combining both, an electric bus engine model and a microscopic traffic simulator. Thus, an integrated tool for analysts to test different set-ups of the electric bus within a controlled environment, and observe its dynamics in urban traffic will be discussed. The following chapter gathers the requirements regarding the combination of different simulation models and describes the application that implements them.

3.2 Integration Requirements Analysis

For the development of a simulation platform that can be used to analyse how the traffic dynamics and the network topology affect the performance of an electric bus a study of the features that the application needs to include, is necessary. Here, the representation of two different systems having different aspects and resolutions is imposed. On one hand, there is the traffic system, that is the road network (expressing the physical infrastructure and the topology) and the vehicle-entities that move on it. On the other hand, the electric bus system is defined in terms of its powertrain subsystem such as the set of battery and traction motor among others.

To address the issues of the traffic system a microscopic modelling approach is required. In these models the level of resolution goes till the individual behaviour of vehicles, the interaction with each other and with the road network. Instead, for the electric bus system a nanoscopic model is more proper to describe the operations of specific parts and processes of the vehicle.

The integration among them is achieved by associating the electric bus powertrain subsystem to a vehicle entity (corresponding to a vehicle of class bus) of the microscopic traffic model. Thus, important criteria for the selection of the simulators (implementing the microscopic and nanoscopic models) are the ease of access to the respective model variables, the application programming interfaces (API) and communication protocols. For example, a simulator that already implements a communication protocol, providing a good API, allows an easier data exchange.

The satisfaction of the aforementioned requirements will provide a flexible distribution allowing a networked access to the simulators core level and thus high-speed data interconnection and exchange between them.

The presented integrated platform might need to support bidirectional communication capabilities. Certainly, the traffic simulator has to provide speed variables to the electric bus powertrain subsystem (EBPS) simulator. Albeit, it is not strictly necessary to have the outputs generated from the EBPS simulator sent to the traffic simulator, this could be important in case of a real-time parameters adjustments of the simulated scenario. However, all of these transactions should occur in the same time step.

Another critical issue that must be properly tackled to achieve the desired goal is the synchronization of simulators. Although traffic simulators are not implemented with hard real-time constraints, the processing power of today's computers allow us to consider that this is quite acceptable.

Therefore, given that most acceptable traffic simulators are prepared to support thousands of individual vehicles in real time, at least a frame rate of 10Hz should be achievable to ensure an efficient data exchanging between simulators. If one wants to perform a simulation with more than one electric bus vehicle, then a larger data flow should be expected between the two simulators. Moreover, all step calculations need to be inferior to the overall frame rate of the simulation for a correct user experience.

It must be noted, that the issues cited above are the most obvious concerning the integration of a traffic simulator with an electric bus engine simulator. However, implementation issues may also arise depending on the architecture and the software chosen.

In the following section an architecture towards the integration of the simulators is proposed.

3.3 Proposed Solution

In this section it will be provided a technical design and solution for the integration of a traffic and an EBPS simulation. Taken into account all issues and requirements stated on the former section, a practical solution is proposed below. Firstly, it is presented the chosen software according to the requirements discussed in 3.2 then, the system architecture is explained as well as the methodology used for its development.

3.3.1 Simulation Package Selection

As pointed out previously, the use of two different simulators may be a feasible approach when simulating electric bus powertrain in an urban context.

In order to implement the physical road infrastructure and the traffic dynamics through vehicular movements in microscopic level resolution, the SUMO software suite has been considered. SUMO is a highly portable, microscopic road traffic simulation package designed to handle large road networks and has a strong commitment with the academia and research community [SUM].

SUMO is not only just a traffic simulation, but rather a suite of applications which help to prepare and to perform the simulation of traffic. As the traffic simulation involves the representation of road networks and traffic demand to simulate in an own format, both have to be imported or generated using different sources [BBEK11]. Although SUMO's network model is quite coarse respect to similar commercial applications, it still provides a fast execution time and its "remote control" interface (TraCI API) for interaction with external applications raises SUMO to be a good candidate for appraising new traffic control algorithms and for net-wide investigations. By implementing different vehicle types, SUMO also allows the simulation of public transport or emergency vehicle prioritization at intersections [KHRW02].

For the simulation of electric bus operations and driving cycle a mathematical model of an EBPS model implemented in MATLAB Simulink framework [PRRA12] has been considered.

This particular model of the EBPS has been chosen due to the fact that is a FEUP's project and thus access to the model's code and data could be achieved. The proposed integrated platform can result in value-added to the R&D project as the relatively new EBPS model can profit by the more accurate analysis in realistic traffic conditions the proposed platform can provide.

As has been mentioned previously, for having a distributed framework is necessary a mechanism that "puts" the different models implemented in different environment communicating between them. That is, is necessary a middle-ware layer that synchronizes the operations among the models and maps the corresponding variables and services of each of them.

One of the goals of this thesis project is to devise a generic framework allowing not only an easy interoperability of different tools but also the reusability of legacy models. Two approaches have been considered in designing the models' communication and integration, one based on an in-house developed solution and the other based on the IEEE HLA standards.

The first approach uses the TraSMAPI layer, which provides an abstraction over the microscopic traffic simulators allowing real-time communication with them.

The second approach follows the IEEE 1516-2010e HLA standards. The main idea behind the implementation of the HLA guidelines is the promotion of the system interoperability and the reuse of legacy software. While in the first case the synchronization of the models is embedded to simulators, with the HLA approach the synchronization is relied on the use of an HLA/RTI middleware (following the HLA standards) and it becomes transparent to the user. For the purpose of the thesis was considered the use of the commercial package Pitch pRTI that implements the above mentioned standards. The advantages of the pRTI over other commercial and free open-source implementations are the extensive documentation and the user friendly interface that the package provides.

3.3.2 Proposed Architecture

In this section an overview of the proposed system based on the HLA architecture (depicted in Figure 3.1) is provided.

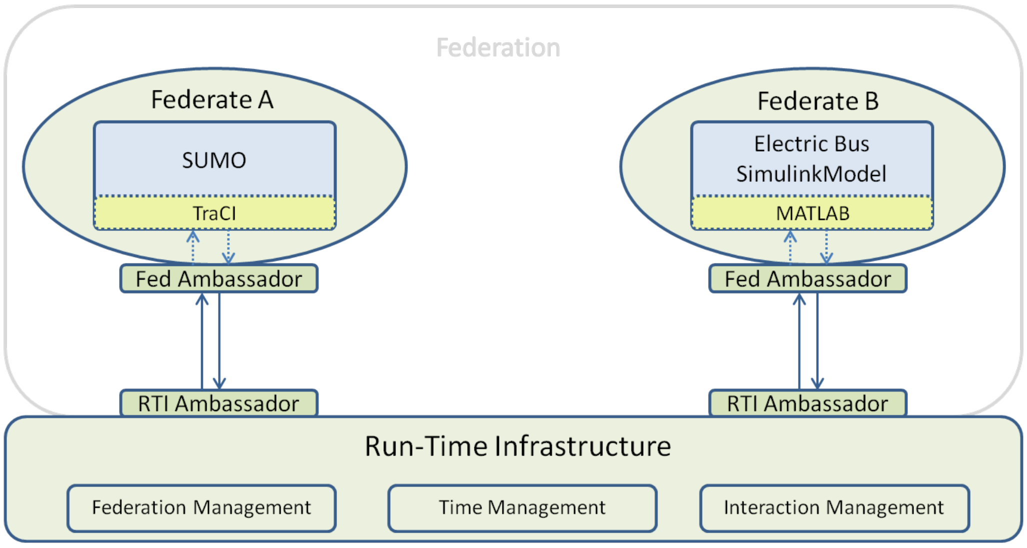


Figure 3.1: System Architecture

This architecture is similar to a simple High Level Architecture one, where four main modules can be identified:

- **Run-Time Infrastructure:** The middle-ware responsible for the management of all simulation process. The RTI supplies services required by distributed executions, it routes messages exchanged and data exchange between the SUMO and the EBPS federates.
- **Federates:** The module corresponding to the HLA compliant simulation entities. For the proposed architecture these are SUMO and EBPS model
- **RTI Ambassador:** Is as specific interface for communication with the RTI. Therefore, it is used whenever one wants to perform calls to the RTI. This interface is already implemented by the chosen RTI, which in this case is the Pitch pRTI.
- **FED Ambassador:** Is a specific interface allowing RTI to communicate with the federate. Unlike the RTI ambassador, this interface shall be implemented by the federate developer and it is code language specific. The RTI will deliver interactions to federate by performing calls to its Federate Ambassador.

> Each federate represents an HLA compliant simulation entity. The compliance is achieved using the federate ambassador interface to exchange data between the RTI and the simulation tool. In this way, the RTI can communicate with the simulation tool through the federate ambassador methods.

Internally each federate comprises a simulation tool, and should ensure the communication between federate ambassador and its tool.

As can be seen in Figure 3.1, the communication between "Federate A" ambassador and SUMO is performed through SUMO's API. In this sense, whenever simulation data are required, the RTI ambassador performs calls to federate ambassador that communicate with SUMO through TraCI.

In a similar way the communication between "Federate B" ambassador and Simulink model is performed through MatLab, the Simulink's API.

3.3.3 Prototype Development Planning

In this section it will be presented the methodology for prototype's development describing the proposed framework. Here, it will be identified the main tasks and described the objectives and methods of each one. The work development was divided into twelve main tasks, each one with its specific objectives described below.

Task 1: Build communication between a Java external application and Matlab

Create a simple Java application that communicates with Matlab, sending it two numbers and get the sum of them as return.

Simple development description: Create sockets to establish the communication between the two applications. Implemented sockets on both Java and Matlab applications as a client-server architecture.

Task 2: Control the Simulink model by an external application through Matlab

Creating a simple Simulink model and control its simulation by an external application using Matlab's methods as API for Simulink.

Simple development description: Create in Simulink model a simple model that receives a number as input and returns twice that number as output. Study what Matlab's methods could be used to control the Simulink model, and run the model step-by-step while introducing news inputs.

Task 3: Design a simple scenario for SUMO

Create a simple scenario for simulation, with one bus vehicle;

Simple development description: Create the necessary files for the road network implementation in SUMO. Also include some bus stops as well as the bus vehicle itself.

Task 4: Build communication between a Java external application and SUMO

Create a simple Java application that communicates with TraCI, to obtain the speed of a car at each time step.

Simple development description: Use Java sockets to establish the communication with TraCI. Then use the necessary methods, provided by this API, to get the speed of a specific vehicle in SUMO simulation.

Task 5: Install the Pitch pRTI

Download and install Pitch pRTI and run the provided example.

Simple development description: This task aims the installation of the RTI and the execution of an example. The Pitch pRTI package download provide set of examples useful to understand the concepts and architectures of federates an its communication methods with the RTI.

Task 6: Create FOM for the federation intended

Create the FOM file needed to the execution of the federation.

Simple development description: Create the Federation Object Model file in order to execute the federation. In this file the Interaction Classes, the Object Classes, the attributes of each class and the variables types should be defined.

Task 7: Create Federate Ambassador for SUMO

Create the Federate Ambassador for SUMO in order to communicate with RTI.

Simple development description: Implement all the needed methods of federate ambassador standards for SUMO's federate.

Task 8: Create Federate Ambassador for MATLAB/Simulink

Create the Federate Ambassador for MATLAB/Simulink in order to communicate with RTI.

Simple development description: Implement all the needed methods of federate ambassador standards for MATLAB/Simulink's federate.

Task 9: Integrate Simulink with SUMO through RTI

Create the federation and run the both, SUMO and MATLAB/Simulink federates.

Simple development description: Execute the RTI, create the federation, execute both federates to join the federation. Execute the simulation and destroy the federation at the end.

Task 10: Execute Simulink model under TraSMAPI control

Run and control the simulation of the same simple Simulink model through TraSMAPI.

Simple development description: Implement in TraSMPI the API methods for Simulink as done to other simulators (SUMO, ITSUMO [TARO12] and AIMSUN).

Task 11: Integrate Simulink with SUMO through TraSMAPI

Run both SUMO (with a simple road network) and Simulink model from TraSMAPI

Simple development description: Create the simulation management to control and coordinate both simulations. Run both simulators from TraSMAPI and execute the integrated simulation.

Task 12: Test and validation

Perform tests to validate de integrated framework.

Simple development description: Perform both under TraSMPAI and HLA simulations and compare the obtained results for HLA integration validation. Run the HLA integration with two different driver behaviour to evaluate the performance of the electric engine in different situations.

3.4 Summary

Correctly defining the requirements was fundamental for identifying the required steps to develop a solution for the integration of a microscopic traffic simulator with an electric bus subsystem simulator. In this chapter, was presented these requirements as well as the proposed solution architecture. Furthermore, a brief explanation of the identified tasks, in order to fully develop the aforementioned architecture, was presented. In the following chapter a more technical overview of the prototype model will be detailed.

Chapter 4

Development Software Overview

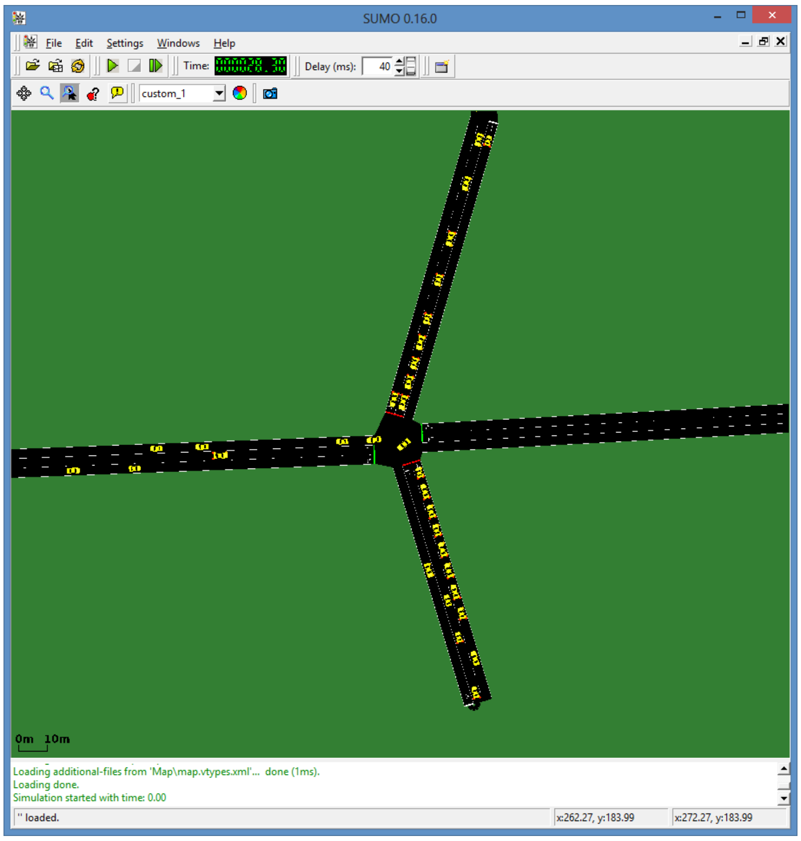
In the former chapter, a comprehensive overview of current issues for the integration of simulators was presented along with the requirements for the solution to address them. This chapter describes the chosen software for development the aforementioned solution. First, the SUMO traffic simulator and EBPS are introduced as the main simulation tools. Finally, TraSMAPI and Pitch pRTI are presented as middle-ware solutions for the integration of both simulators.

4.1 SUMO Microscopic Traffic Simulator

SUMO (Simulation of Urban MObility) is a well-known open-traffic microscopic traffic simulation package that appears for the first time in 2001. Today SUMO is already in its 0.16 version and is one of the most commonly used microscopic traffic simulator in the research community. In this chapter, a brief description of SUMO is presented as well as an brief overview of its architecture. Furthermore, it is made a description of necessary SUMO's network files in order to perform a traffic simulation.

SUMO became an important tool in many researches within urban traffic and transportation domain. It have gained its place among the academic community with many of scientific papers referring to it. Today, SUMO is not just a traffic simulation, but rather a suite of applications which help to prepare and to perform the simulation of traffic. In fact SUMO has been used in several research topics such as route choice [DPW10], traffic light algorithm [MSTR12, GNAO12, KBM$^+$05] or simulating vehicular communication [LC08, DSN08, LTV$^+$06], among others. Figure 4.1 illustrates a typical simulation scenario on SUMO.

SUMO is a pure microscopic traffic simulation. At each vehicle is associated an identifier, the departure time, and the vehicle's route through the network. It is also possible to describe each vehicle in more detail. For example, it is possible to define arrival properties of the vehicle, which lane it could use, its velocity and its position. Also, each vehicle can be assigned to a vehicle-type which describes its physical properties and variables of the used movement model. This is an important feature for this work since it allows us to make an one to one association between the vehicle bus in SUMO and the EBPS of the electric bus. Sumo follows time-discrete simulation



```
Loading additional-files from 'Map\map.vtypes.xml'...  done (1ms).
Loading done.
Simulation started with time: 0.00
'' loaded.    x:262.27, y:183.99    x:272.27, y:183.99
```

Figure 4.1: Example of the Simulation of Urban Mobility (SUMO) traffic simulator interface

having a simulation step of 1 second by default. It is also space-continuous and internally, each vehicle's position is described by the lane the vehicle is on and the distance from the beginning of this lane.

As the traffic simulation SUMO requires the representation of road networks and traffic demand to simulate in an own format, both have to be imported or generated using different sources. Regarding the network, there are different ways for creating road networks for SUMO. It can be either generated using an application named *netgen* or generated by importing a digital road map. The road network importer *netconvert* allows to read networks from other traffic simulators as VISUM, Vissim, or MATsim. It also reads other common formats, as shape-files or Open Street Map [KHRW02].

With respect to the traffic demand, there are some applications for SUMO that allow the generation of traffic flux over the networks. *jtrrouter* is a route computation application that uses definitions of turn percentages at intersection for computing routes through the network. Such an approach can be used to set up the demand within a part of a city's road network consisting of up to ten nodes. A further application, *dfrouter*, computes routes by using information from loop detectors. This approach is quite successful when applied to highway scenarios where the road network does not contain rings and the highway entries and exits are completely covered by detectors [BBEK11].

In 2006 the simulation was extended by the possibility to interact with an external application via a socket connection and thus, an API for SUMO was developed. TraCI (Traffic Control Interface) is an API for SUMO that gives the access to a running road traffic simulation, it allows to retrieve values of simulated objects and to manipulate their behaviour.

TraCI have a extensive documentation of the methods for communication with SUMO. It is composed by three main sets of functions which are related to the information access, to the states change of and to the subscription of determined structure's variables.

4.2 EBPS - MATLAB/Simulink

In this section, it is explained the simulation system EBPS and its characteristics. However, to a better understanding of the electric bus powertrain subsystem, it is important to know some concepts of the framework where it was developed. Therefore, an overview on the MATLAB tool as well as the Simulink environment is firstly presented.

MATLAB

MATLAB (MATrix LABoratory) is a high performance software for numerical computation, data visualization and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. Is an interactive system and a programming language for computing technical and scientific cooperation in general [Mata]. MATLAB, allows to analyse data, develop algorithms, and create models and applications. Its built-in maths functions permit the resolution of many numerical problems faster than with traditional programming languages, such as Fortran, Basic, C/C++ or Java.

Moreover, the solutions of the problems are expressed almost exactly as they are written mathematically [Gui98]. MATLAB has evolved over a period of years with input from many users. In university environments, it is the standard instructional tool for introductory and advanced courses in mathematics, engineering, and science. In industry, MATLAB is the tool of choice for high-productivity research, development, and analysis. MATLAB features a family of application-specific solutions called toolboxes, very important to most users of MATLAB.

Toolboxes allow to learn and apply specialized technology. Toolboxes are comprehensive collections of MATLAB functions (M-files) that extend the MATLAB environment to solve particular classes of problems. Areas in which toolboxes are available include signal processing, control systems, neural networks, fuzzy logic, wavelets, simulation, and many others [Wha].

Simulink

Simulink is a software package for modelling, simulating, and analysing dynamical systems. It supports linear and non-linear systems, modelled in continuous time, sampled time, or a hybrid of the two. For modelling, Simulink provides a graphical user interface (GUI) for building models as block diagrams, using click-and-drag mouse operations(see Figure 4.2). With this interface there is no need to formulate differential equations and difference equations in a language or program like previous simulation packages [SN93].

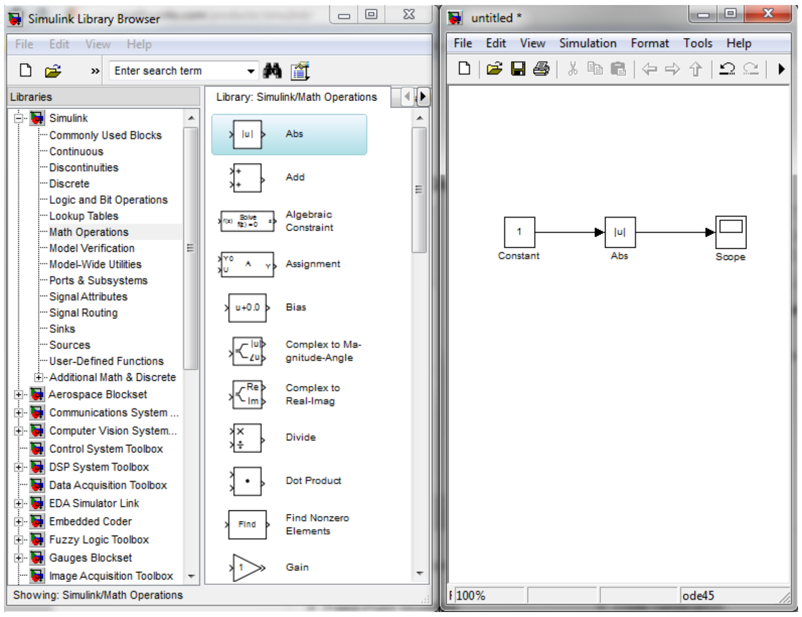


Figure 4.2: Simulink Graphical User Interface

Simulink includes a comprehensive block library of sinks, sources, linear and non-linear components, and connectors. It allows customization and creation of new blocks. Models are hierarchical, so one can build models using both top-down and bottom-up approaches. One can view the system at a high level, then double-click on blocks to go down through the levels to see increasing levels of model detail. This approach provides insight into how a model is organized and how its parts interact. After defining a model, it is possible to simulate it, using a choice of integration methods, either from the Simulink menus or by entering commands in MATLAB's command window [Mat99].

The menus are particularly convenient for interactive work, while the command-line approach is very useful for running a batch of simulations. Using scopes and other display blocks, it is possible to see the simulation results while the simulation is running. In addition, parameters can be changed and immediately see what happens, for "what if" scenarios exploration. The simulation results can be put in the MATLAB workspace for post processing and visualization. Model analysis tools include linearisation and trimming tools, which can be accessed from the MATLAB command line, plus the many tools in MATLAB and its application toolboxes. And because MATLAB and Simulink are integrated, it is possible to simulate, analyse, and revise models in either environment at any point [DH01].

One of the key features of Simulink is that it is built on top of MATLAB. As a result, Simulink users have direct access to the wide range of MATLAB-based tools for generating, analysing, and optimizing systems implemented in Simulink [Mat99].

EBPS (Electric Bus Powertrain Subsystem)

The Electric Bus Powertrain Subsystem (EBPS) is a mathematical model of a electric bus subsystem implemented in MATLAB Simulink [PRRA12]. This model has several subsystems, which are used to calculate specific parameters. One of these subsystems represents the vehicle's powertrain, taking into account the forces that work against its movement and the gear ratios involved. An output of this subsystem computes the amount of required energy for a driving cycle to be completed. There is a third subsystem that calculates the amount of energy that may be possibly recovered from the regenerative braking, taking into account the kinetic energy of the vehicle. The two other subsystems are related to the batteries and the super-capacitors, evaluating whether they are capable of absorbing the energy from the braking [PRRA12]. Figure 4.3 illustrates the main subsystem of the model.

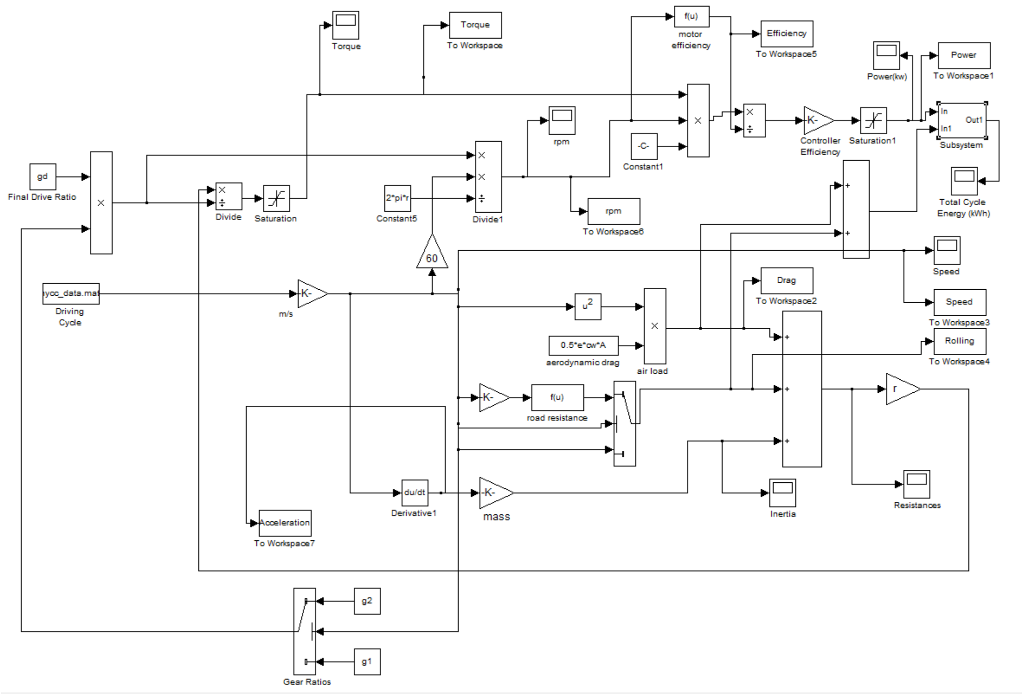


Figure 4.3: Main subsystem of EBPS model

This system is modelled in continuous time. It receives a vectorial structure as input, where velocities are related with time instants. As outputs, the model produces a structure for each metric, with values associated to an instance of time. The most significant parameters calculated by this model considering are:

- **Power:** Expresses the power required for each instant of the driving cycle.
- **Total Energy Cycle:** The energy required to complete a whole cycle.
- **Kinetic energy:** It is known that the effect of the vehicle mass when accelerating and stopping the vehicle in urban conditions has considerable influence

on vehicle performance. So, the kinetic energy is used in this model for calculating the amount of energy dissipated in braking, considering the tires and air resistance.

- **Battery Charging:** One of the main assessments is whether the lithium-ion batteries are able to absorb the burst of energy that a braking can cause. Usually, this energy is converted into heat and dissipated through the brake system. Thus, this parameter is used to calculate how must energy could be recovered by braking system.
- **Super-capacitors:** Used to investigate whether super-capacitors are able of absorbing regenerative braking energy.

In order to be used in the intended integration, this model needs to be modified to a discrete model. In this way, it could receive a unique variable and produce output values for that specific instance. In the following chapter (see Chapter 5) it will be presented the performed modifications to the model.

4.3 TraSMAPI

For the first integration approach, it was chosen an ad-hoc connection of the simulation models. In this context TraSMAPI (Traffic Simulation Manager Application Programming Interface) offered such solution. TrasMAPI is a tool for the simulation of dynamic control systems in road networks, with special emphasis on Multi-Agent Systems. This tool allows real-time communication with microscopic simulators providing a framework for the development of multi-agent solutions [TARO12]. The abstraction over the simulator enables to run different traffic simulators using the same API (Application Programming Interface) allowing, for example, the comparison of results of the same application in different simulators [TARO10].

The TraSMAPI architecture is based on three main Modules: the Communication Module, the Statistics Module, and the Multi-Agent System Framework (see Figure 4.4). These independent modules have a specific and well-defined function in the whole system and interact with each other to form the whole solution. The user only needs to implement the agents and choose a simulator supported by the API TraSMAPI, to run the simulation.

In order to turn the agent's behaviour transparent to the simulator in use, they are used Java objects that provide the abstraction of several methods. However, this abstraction is only certain if the simulator in use implements all the functionality expected for the agents, for example, a simulator can implement the roads density sensor while the other simulator cannot do it [TARO12].

The most important TraSMAPI modules are Communication and Statistics. The Communication module provides an abstraction layer responsible for interaction and communication via sockets with microscopic simulators, while Statistics is responsible for storing all the information transmitted to the multi-agent system simulator. This multi-agent system, TraSMAPI MAS (TraSMAPI Multi-Agent System) is yet another module that is providing by TraSMAPI.

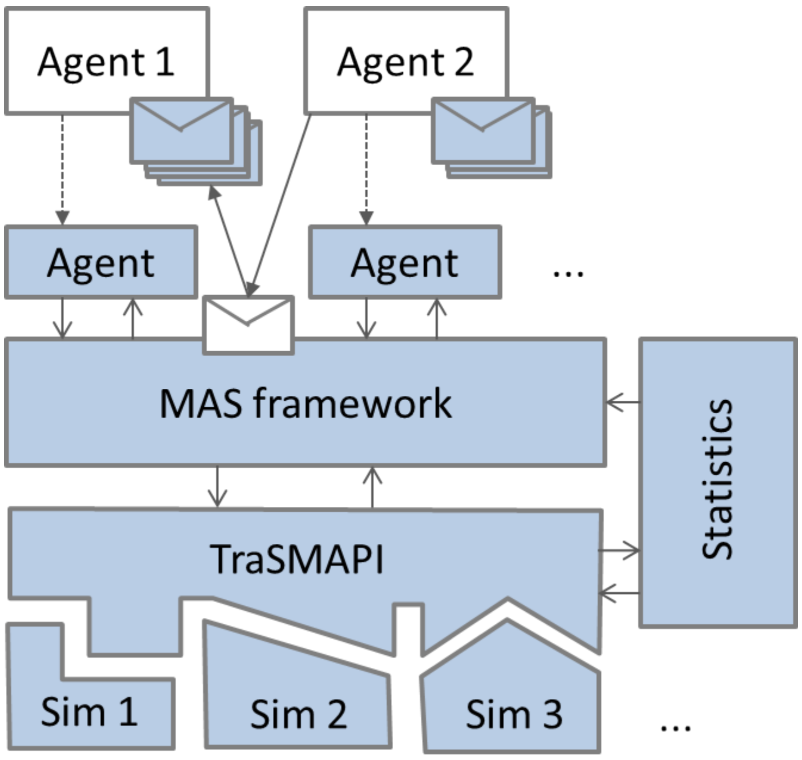


Figure 4.4: Modular structure of TraSMAPI and overall architecture of MAS (from [TARO12])

The communication in the Multi-Agent System Framework is based on an asynchronous message system. In this system, messages from agent to agent are supported, as well as broadcast communication. MAS not only directly accesses the Statistics module but also controls the activity between the system, since they are created according to a basic interface that already implements all interactions with the multi-agent system, such as messages exchange or an action request from the agent in each simulation step, which in turn guarantees the timing.

Each agent is able to use all of the API objects that are proxies for existing objects in the simulation itself, i.e. these objects are an abstraction that allows the manipulation and extraction of transparent information to the programmer of the associated element in the simulation [TARO12, TARO10].

However, the importance of TraSMAPI for this project is mainly related with the communication module that it provides. TraSMAPI will be mainly used as a first approach on the integration of the both systems SUMO and EBPS. In this sense, its abstraction layer for interaction with microscopic simulators brings great value for the integration, allowing an easier control of the simulations.

4.4 Pitch pRTI

For the second type of integration it was decided to follow an industrial standard approach using the HLA concepts based on the IEEE 1516-2010e standard. There are currently different available HLA/RTI implementations, and the Pitch pRTI was the one used for this project.

The main purpose of HLA is to promote reusability and interoperability of simulations. It was developed to satisfy the requirements of simulations in a wide variety of areas including analysis, testing, training, and other engineering functions.

An HLA federation is primarily comprised of one or more federates, the federation object model and the Runtime Infrastructure. HLA is a software architecture that permits objects in one simulation to exchange data with objects in another simulation through services provided by the RTI. Interactions between federates are not conducted directly, but through the functions provided by the RTI. In addition, the RTI carries out support services required for federation management. Thus, the RTI is a distributed run-time interface for the whole federation [KDW00].

There are currently different available RTI implementations, and the Pitch pRTI was the one chose for this project. Pitch pRTI is a commercial Run-Time Infrastructure that already implements the HLA evolved standards. These new standards contains the same functionality as HLA 1516-2000 but it also provides the new C++ and Java APIs for a large number of operating systems together with an implementation of a majority of the new HLA Evolved functionality features [MML$^+$08], for example:

- Fault tolerance support services
- Web Services support/API
- Modular FOMs
- Smart Update rate reduction
- Encoding helpers
- Extended XML support for FOM/SOM, such as Schemas and extensibility

One of the advantages of this RTI is the support provided, allowing a greater ease on the use of it and a better understanding the HLA concepts. In fact, the Pitch pRTI provides not only a user-friendly graphical interface, but also a sample of federates and FOMs. Figure 4.5 shows the RTI graphical interface.

The sample includes one car simulation federate, one federate for visualization and another one for managing the simulation. The objective of the sample federation is to see the consumptions of different car models and different types of fuel. Figure 4.6 illustrate the execution of this sample federation.

In the upper left corner is represented the federate responsible for the visualization of the simulation. Here it could be seen the current fuel level and the position of each car on the map. In the upper right corner is illustrated the federate manager. This federate are responsible for managing the federation like start and stop simulation and load scenario. Furthermore, in the

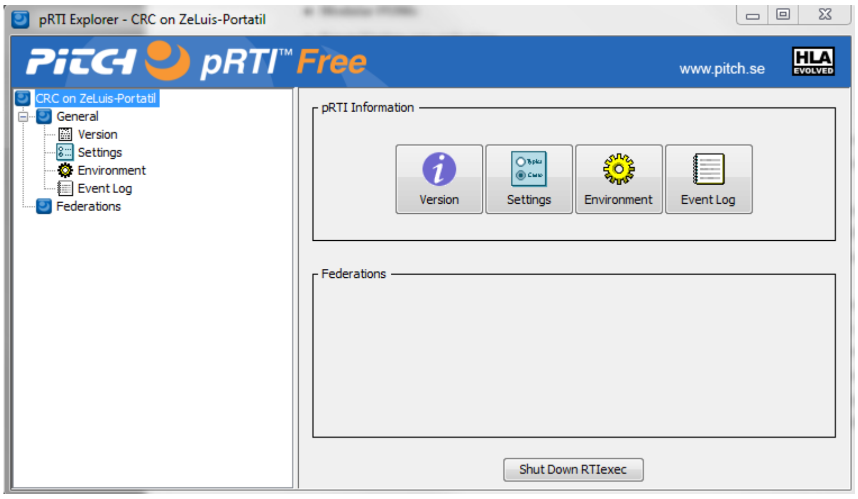
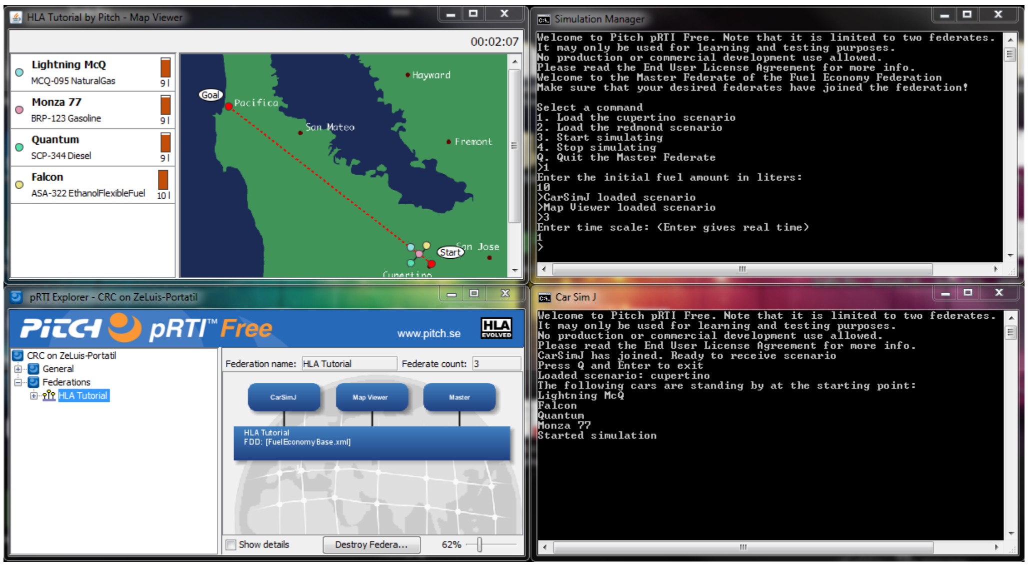


Figure 4.5: Pitch pRTI Graphical Interface

lower right corner is illustrated the car simulation federate, which is responsible for simulating all the cars for the federation. For last, in the lower left corner is represented the RTI graphical interface. Here it can be seen the federation and the federates associated to it.



```
Welcome to Pitch pRTI Free. Note that it is limited to two federates.
It may only be used for learning and testing purposes.
No production or commercial development use allowed.
Please read the End User License Agreement for more info.
Welcome to the Master Federate of the Fuel Economy Federation
Make sure that your desired federates have joined the federation!

Select a command
1. Load the cupertino scenario
2. Load the redmond scenario
3. Start simulating
4. Stop simulating
Q. Quit the Master Federate
>1
Enter the initial fuel amount in liters:
10
>CarSimJ loaded scenario
>Map Viewer loaded scenario
>3
Enter time scale: (Enter gives real time)
1
>
```

```
Welcome to Pitch pRTI Free. Note that it is limited to two federates.
It may only be used for learning and testing purposes.
No production or commercial development use allowed.
Please read the End User License Agreement for more info.
CarSimJ has joined. Ready to receive scenario
Press Q and Enter to exit
Loaded scenario: cupertino
The following cars are standing by at the starting point:
Lightning McQ
Falcon
Quantum
Monza 77
Started simulation
```

Figure 4.6: Sample Federation Execution

In this chapter was described the software used in this project highlighting the main features and its importance to the works development. In the following chapters a more technical overview of the prototype model will be detailed.

Chapter 5

Implementation

Following the main tasks identified and presented in Section 3.3.3, this chapter describes the implementation process towards the coupling of the microscopic traffic simulation SUMO with the MatLab/Simulink electric bus model.

5.1 Communication Modules

Integrate different simulation tools, allowing data exchange between them, requires them to be prepared to communicate with external applications. Considering this, the first step in the implementation process was to create the communication modules for MatLab/Simulink and SUMO.

5.1.1 MatLab/Simulink Module

As it has been said before, MatLab could be seen as an API for Simulink since Simulink models can be controlled by MatLab methods. In this sense a communication interface was implemented for Matlab. Simulink models are standalone models and exist in symbiosis within the MatLab environment. The only way to access them externally is through MatLab's methods and calls [Mata]. For this reason a "control" using MatLab's interface to Simulink needs to be applied. A communication channel using one of the MatLab's interfaces must be implemented. It follows a description of the necessary steps to enable the communication between SUMO and Simulink.

In the next paragraphs for the sake of example and clarity is explained how an external application communicates with MatLab and SUMO. After this tutorial the core of the implementation is discussed as well as the way the chosen middlewares have been used.

5.1.1.1 Communication between Java external application and MatLab

First, it was developed a simple Java application in order to create and validate the communication interface with Matlab. This simple program should send two numbers to Matlab which should send back the sum of them as result.

```java
1  double first = 2;
2  double second = 3;
3
4  MatlabProxyFactory factory = new MatlabProxyFactory();
5  MatlabProxy proxy = factory.getProxy();
6
7  proxy.setVariable("a", first);
8  proxy.setVariable("b", second);
9
10 proxy.eval("c = a + b");
11
12 double result = ((double[]) proxy.getVariable("c"))[0];
13
14 System.out.println("Result: " + result);
15
16 proxy.disconnect();
```

Listing 5.1: Code example for create functions for word counting

In a first approach, it was thought to use sockets to establish the communication with Matlab, but since the program is written in Java, the *matlabcontrol* API was used. The *matlabcontrol* is a Java API to interact with MatLab allowing for call MatLab methods from Java [Matb].

Using this, it is only necessary to create a proxy to work as an image of the MatLab application. After that, all the calls to MatLab are performed through this proxy. In the Listing 5.1 a script of the simple Java application uses the API to communicate with MatLab is shown.

Looking at the script, in the lines 4 and 5 can be seen how the MatLab proxy is created (The MatLab instance is created when the command of line 5 is performed). Furthermore, in lines 7 e 8 is exposed how to set variables while in line 12 is shown how to get the value of a variable from MatLab. Last, in the lines 10 and 15 is demonstrated how to perform a MatLab command from Java and how to destroy the MatLab proxy.

5.1.1.2 Control of Simulink model simulation through MatLab

After the development of the communication with MatLab the next step is to create and execute a simple Simulink model and control its simulation step-by-step through Matlab. Matlab provides at least 3 ways to execute a Simulink model which are *sim()*, [Mata], or [Mata] functions. Each one of these functions has its advantages and disadvantages depending on mode the model operates. These advantages and disadvantages are illustrated in Table 5.1.

Among the aforementioned modes the *set_param()* function was chosen as it fulfils the requirement of external step-by-step

After that, it was necessary to create a simple Simulink model to test and validate the simulation control through Matlab commands. This model is able to read two numbers from Matlab variables as input and save the sum of them as result into another Matlab variable. Figure 5.1 illustrate this simple model.

Function	Advantages	Disadvantages
sim()	Can choose any solver from Simulink model. Can use standard Simulink In and Out ports.	Very slow. Does not allow to control the simulation step-by-step.
model()	Very fast. Can use standard Simulink In and Out ports	Need to create own ODE solver. Output data collected only at evaluation step resolution. Ignores all of the solver configuration parameters.
set_param()	Can choose any solver from Simulink model. Output data can be at a higher resolution than evaluation steps. Moderately fast. Provide intuitive mechanisms to control the simulation.	Must create S-function model input and model output ports. Cannot run MatLab headless.

Table 5.1: A comparison between different functions for execute Simulink models simulation

The input and output blocks use *S-Funtions* in order to be possible to use the *set_param()* method as seen in the Table 5.1. The implementation code of the S-Functions can be found in Appendix A.

Once the model was created, a set of commands is performed in MatLab command window in order to test the simulation control.

```
a = 2;

set_param(example.mdl,'SimulationCommand','start');
set_param(example.mdl,'SimulationCommand','pause');

for j=1:4,
b=j;
set_param(example.mdl,'SimulationCommand','step');
c;
end

set_param(example.mdl,'SimulationCommand','stop');
```

Listing 5.2: Script to control a simple Simulink model through MatLab

First, a simple command was used to set the variable that will be used by one of the input blocks of the Simulink model. Then, the *set_param()* command was used to start and immediately pause the simulation. By doing this, the simulation remain paused at the beginning without perform any calculation. After that, a cycle is used to test different inputs for the second variable. In this cycle, after set the second variable, the *set_param()* was used again, but this time to advance one step in the simulation. This will perform one cycle through the model and pause automatically at the beginning. At this point, the variable to where the output block saves the result, as

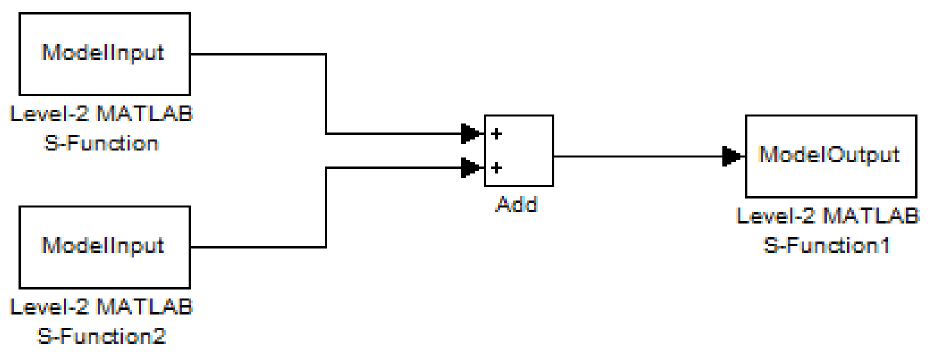

ModelInput
Level-2 MATLAB S-Function
ModelInput
Level-2 MATLAB S-Function2
+
+
Add
ModelOutput
Level-2 MATLAB S-Function1

Figure 5.1: Simple Simulink Model

already been instantiated with the result of the sum of the input values. Each cycle performed in the cycle "for" corresponds to one step in the simulation of the model. At the end, the *set_param()* command is called again but to stop the simulation. The script of the whole process just described is presented in the Listing 5.2 where "a" and "b" corresponds to the input variables and the "c" correspond to the output one.

5.1.1.3 Control Simulink model simulation through Java External Application

At this point, the communication between the Java application and MatLab is established and the Simulink model simulation can be controlled by MatLab commands. The final step is the creation of the communication module to allow an Java application to control the simulation of the Simulink model.

The module is composed by a set of higher level functions implemented in Java, that comprises the necessary commands to communicate with MatLab in order to control the simulation. The implemented functions were the follows:

- launch()
- connect(String model)
- simStep()
- send(String command)
- getVariable(String variable)
- close()

The first function is the function launch() and is used to execute the MatLab application. This function creates the MatLab proxy with *matlabcontrol* API in order to run MatLab and establish communication with it. The function connect() receives a Simulink model's name as an argument and begins the simulation of that model using the *set_param()* function. The *simStep()* function, as the name suggests, advances by one step the simulation, using fo that the *set_param()* command.

The other two functions are used to send commands to MatLab (for example, to instantiate a Matlab variable) and to receive MatLab values. The last function is the function close() that is used to stop the simulation and close the connection.

At this point, the MatLab/Simulink communication module is created, which allows any application written in Java to easily control and access the simulation of a Simulink model.

5.1.2 SUMO Module

Unlike Matlab / Simulink, SUMO already comes with an API that provides a communication protocol. In fact, the TraCI interface provides a set of methods that allow an easy interaction with the simulator's state variables. In this sense, it is only needed the development of a communication module for Java applications to interact with TraCI.

5.1.2.1 Communication between Java external application and SUMO

To allow interaction between sumo and the Java application an implementation of a communication module is necessary. This module is composed of a set of functions, implemented in Java, that comprises the necessary commands to interact with SUMO. The following three functions are relative to the connection establishment between SUMO and the Java application:

- launch(String config)
- connect(int port)
- close()

The launch function uses the *ProcessBuilder* Java class to initialized SUMO. It receives, as argument, a string with the path to the configuration file of SUMO. This configuration file has specifies the network, the routes and the port number through which the communication with the external application will be performed. The connect function is used to start the communication channel with SUMO. This function receives the port number as argument(the same port specified in SUMO's configuration file) in order to create the socket and establish the communication with SUMO. The last function stops the simulation, closes the communication channel and closes SUMO.

One must note that the described functions do not uses the TraCI methods, since they do not interferes with the simulation. These functions are only responsible to run the SUMO simulator, to establish the connection with it and to terminate all the process. The functions to control and access to the simulation are described below:

- simStep()
- setSpeed(string id, double speedP)
- getSpeed(string id)

The *simStep()* function is used to advance one step in the SUMO simulation, while the *setSpeed()* and *getSpeed()* functions are used to get and set the actual speed of a specific vehicle. This function receives the id of the vehicle as argument. TraCI has an extensive number of methods, each one relative to an entity of simulation. For the scope of this project, the only TraCI's methods that were used are relative to the vehicle entity and its speed.

With the communication module implemented, a simple program was created to test and validate its functionality. For this purpose, a simple network was used in SUMO to experiment the communication with the external application. The Figure 5.2 illustrate the main interactions between the Java application and SUMO.

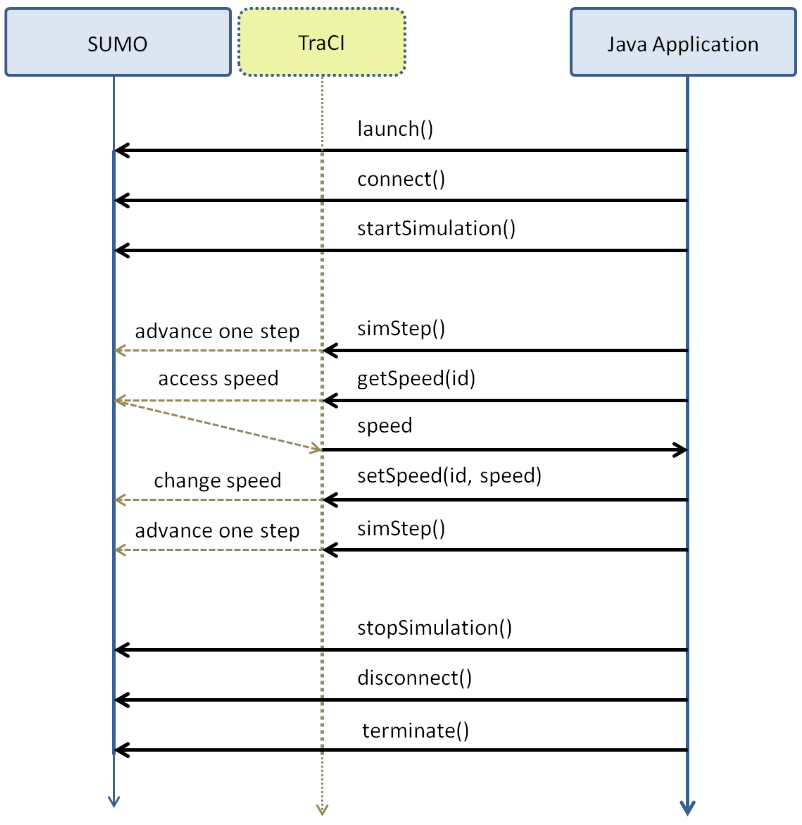


Figure 5.2: Interaction diagram of the test example

The test application consists in a simple connection with SUMO and some interactions with it. First, SUMO is instantiate through the launch function. Then, the application tries to connect with the simulator and ask it to start the simulation. After starting the simulation, the application sends the order to advance one simulation step and asks for the actual speed of a specific vehicle. After receiving the information, the application tries to change the speed of the same vehicle and, again, sends the order to advance one simulation step. With these interactions was possible to verify the validity of the communication module. Finally, the necessary functions are called in order to stop the simulation, to disconnect from SUMO and to close the application.

In this section, have been described the implementation of the communication modules for MatLab/Simulink and SUMO. Furthermore, the test examples have been presented in order to a comprehensive validation of the implemented modules. Next, it will be described the development of the integration of the two systems.

5.2 First approach on Integration

After creating the communication modules, the next step is to integrate the two different systems, combining the different capabilities of each one. Thus, the EBPS could be subjected to the dynamics of an urban traffic environment in terms of acceleration, and frequent stop-and-go episodes.

The first step towards the integration is the adaptation of the EBPS model. As a matter of fact the EBPS Simulink model was devised as a standalone application by its authors. In order for the model being able to be accessed through the function *set_param()*, the EBPS input/output blocks need to be replaced by the input/output blocks of an S-function. Another important aspect to be considered is that the EBPS is a continuous-time model. Therefore, modifications on some blocks were made to turn the EBPS into a discrete time model (e.g. change the derivation block for a difference block). In this way the simulation could be performed step-by-step instead of being executed at once.

Before proceeding to the integration within the standards of the HLA, a simple test application allowing the data exchange between the two simulation tools was developed for first approach.The communication modules have already been tested separately and it has been shown that are capable of controlling and accessing data of each simulation tools. Now, a single program is implemented for the control and access of both simulators.

However, as explained before, TraSMAPI not only provides an API for micro-simulation but also provides an API for simulation control. This way, instead of creating the entire control application for the integrated simulation, it was used TraSMAPI for the simple integration test.

5.2.1 TraSMAPI Integration

To perform the integration, the functions of the communication modules were used to implement the TraSMAPI's interface. In this way, although each simulation tool has its own implemented methods, the external application could control and communicate with both simulation tools using TraSMAPI generic interface. Figure 5.3 shows the integrated architecture with TraSMAPI.

To conclude this integration, a simple program for the external application was implemented. The main steps of the program are:

1. Launch the simulators, and establish a connection with them.
2. Start the simulation on both systems at the same time, and keep them waiting.
3. Advance one step simulation.
4. Get the speed of a SUMO vehicle,send him to the EBPS simulation.

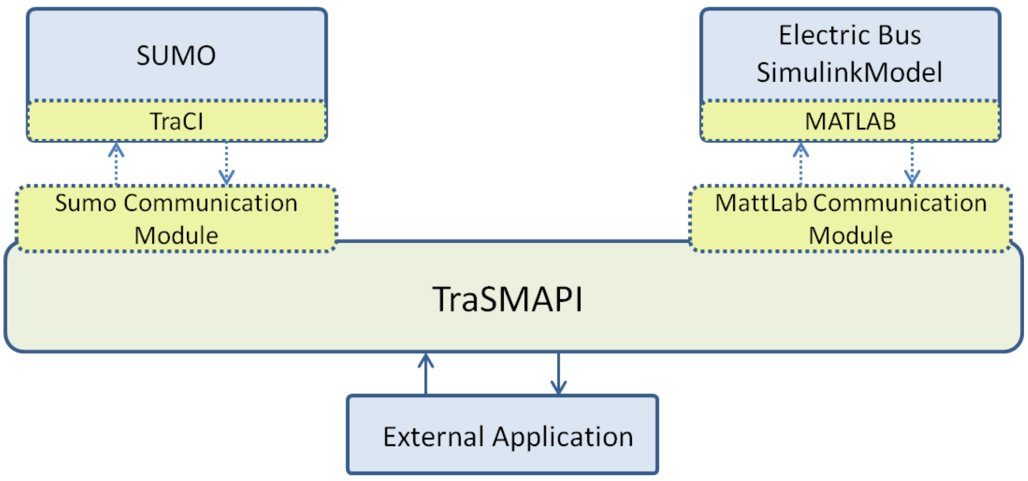

SUMO
TraCI
Electric Bus SimulinkModel
MATLAB
Sumo Communication Module
MattLab Communication Module
TraSMAPI
External Application

Figure 5.3: Integrated architecture within TraSMAPI

5. Get the EBPS outputs.
6. Advance one step simulation.
7. Stop both simulations and disconnect from respective simulators.
8. Close simulators.

The steps 4 to 6 are performed in a cycle till the last simulation step.

5.2.2 Performed Testes

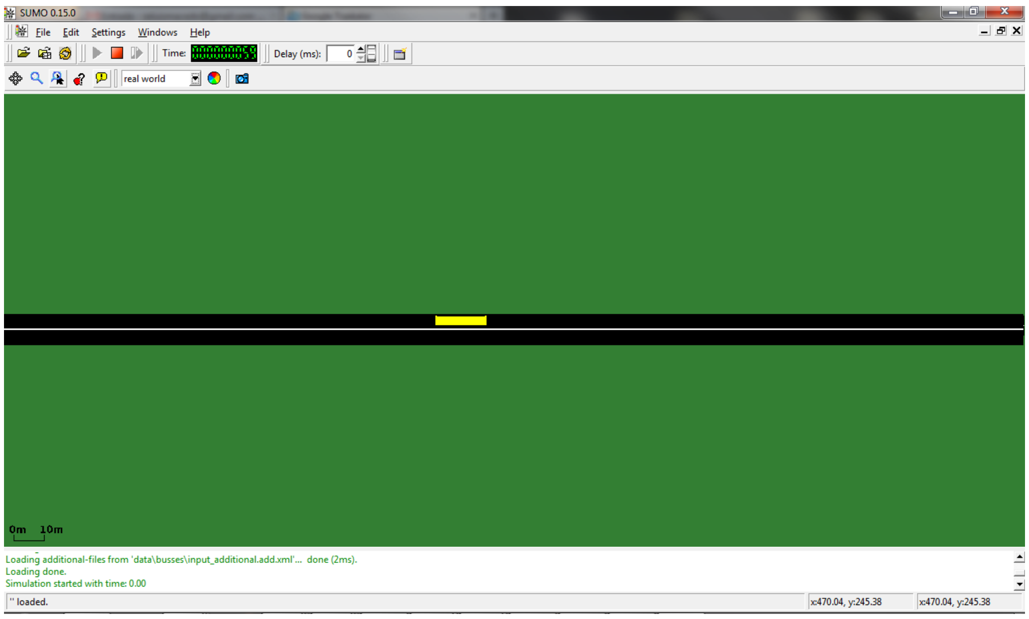
With the first approach of integration completed, some tests were performed to analyse its validity.

For this purpose, the velocities of the field experimental data used to validate the electric bus powertrain subsystem were reproduced in SUMO for the vehicle speed behaviour.

Thus, the outputs from EBPS can be compared with the expected outputs for those specific velocities inputs.

Thereby, a simple road network was design in order to perform such experiment. Since the car velocity will be controlled by the external application at each step, the network cannot comprise situations that might force the car to change its speed without the application order. In this sense, the road network created is compose for just one road and just one car(see Figure 5.4). Thus, the car will not be subjected to traffic situations during the simulation.

After the performed test, the outputs from the EBPS model were analysed to validate the integration. The results shown that for the same inputs used in the field experiment, the outputs were also the same. Therefore, the integration with the TraSMAPI layer was validated. (see Appendix B for the field experiment results)



```
Loading additional-files from 'data\busses\input_additional.add.xml'...  done (2ms).
Loading done.
Simulation started with time: 0.00
'' loaded.    x:470.04, y:245.38    x:470.04, y:245.38
```

Figure 5.4: Interaction diagram of the test example

5.3 HLA based Integration

Having successfully integrated the two systems in a kind of ad-hoc way, this section comprehends the final road towards the coupling of both SUMO traffic simulator and EBPS model. Having addressed all particular issues concerning communication modules and integration aspects, a comprehensive analysis on the HLA components implementation is performed. The main challenge to integrate the systems in the standards of the HLA is the creation of the FOM and the implementation of federates.

5.3.1 Federation Object Model (FOM) Specification

When connecting several simulation systems it is necessary to decide exactly how federates are exchanging data. One important part of this is a description of information that needs to be exchanged.

The FOM is a file that contains a description of the data exchange in the federation, for example the objects and interactions that will be exchanged. This can be seen as the language of the federation. It is required different FOMs when running different simulations since it is needed to exchange data about different concepts.

In the creation of the FOM, three of the most important issues to be considered are the Object classes, the Interaction classes, and the Data types.These entities have to be well defined in order to describe the information that needs to be exchanged between federates.

For the intended simulation, all data that needs to be exchanged are related to bus attributes. In this sense, the object class that needs to be described in FOM is the class Bus. Moreover, to define the class object, all the attributes and its variable types have to be defined too. In the Table 5.2 is represented the table of the object classes for the federation FOM intended .

Attribute	**Type**	**Publish Subscribe**
Name	HLAunicodeString	PS
Velocity	VelocityFloat64	PS
Acceleration	AccelerationFloat64	PS
Power	PowerFloat64	PS
Torque	TorqueFloat64	PS
Efficiency	EfficiencyFloat64	PS
TotalCycleEnergy	TotalCycleEnergyFloat64	PS
BrakingKinectEnergy	BrakingKinectEnergyFloat64	PS
BrakingResistanceEnergy	BrakingResistanceEnergyFloat64loat64	PS
SuperCapacitorsChargingEnergy	SupercapacitorsChargingEnergyFloat64	PS
SuperCapacitorsDischargingEnergy	SupercapacitorsDischargingEnergyFloat64	PS
BatteriesChargingEnergy	BatteriesChargingEnergyFloat64	PS

Table 5.2: Bus Object Class Representation

This class has twelve attributes each one with a specific type. The attribute *Name* was included to identify the bus instance in case there is more than one bus. The velocity attribute is the variable that the traffic simulator will update and the others attributes are the variables for the outputs of the EBPS model. The PS in the third column of the Table 5.2 means that the attribute could be Publish and Subscribed. There are also additional property attributes not described. All the properties are shown in Appendix C.

Regarding the attributes types, the *HLAunicodeString* is already a default HLA type. However, the other types are not, and need to be described in FOM. Listing 5.3 shows an example of the description of a type in FOM.

```
<dataTypes>
    <simpleDataTypes>
        <simpleData>
            <name>VelocityFloat64</name>
            <representation>HLAfloat64BE</representation>
            <units>Litres</units>
            <resolution>0.000001</resolution>
            <accuracy>0.000001</accuracy>
            <semantics>Double that describes the velocity.</semantics>
        </simpleData>
    </simpleDataTypes>
</dataTypes>
```

Listing 5.3: Code example of FOM attribute type definition

To describe a new type, it is necessary to define the default HLA type that it comprises, the units that it represents, the resolution that it accepts and a simple description about its semantic.

To conclude the FOM description, the interaction classes must be described. An interaction is something that does not persist over time, it is used to perform some immediate reaction. The interaction classes created are the Start and Stop interactions. These interactions are used to start and stop the simulation execution on both simulators. Thus, when federates receives one of these interactions, they starts or stops the simulation respectively.

Also, an interaction could have some parameters as arguments, and if it is the case, they need to be specified too. In this case, only the Start interaction comprises an argument. This parameter is used to specify the frame rate at which the simulation should be processed. Tables 5.3 and 5.4 illustrate the interaction classes and their parameters respectively (the specification of the *ScaleFactorFloat32* type can be seen at the FOM script in the Appendix C).

Interaction Class	**Publish Subscribe**	**Parameter**
Start	PS	TimeScaleFactor
Stop	PS	-

Table 5.3: Interaction Classes

name	**dataType**	**semantics**
TimeScaleFactor	ScaleFactorFloat32	How fast will the simulation run compared to real time. Example: 1.0= real time, 2.0 indicates that the simulation runs at twice the speed.

Table 5.4: Interaction Class Parameter

5.3.2 Federates Specification

After the FOM specification being complete, the next step was the development of the federates entities. Unlike the first integration experience, where a unique application had access to the simulation tools through the TraSMAPI communication layer (see Section 5.2), in HLA, each federate application is seen as an independent application. In this sense, each federate was carefully design and developed. Figure 5.5 illustrates the main federate components and interactions.

While developing the federates applications, two aspects had to be taken into account: one is the communication module with the simulators, the other is the federate ambassador module for communications with the RTI.

The federate ambassador is the module through which all the communication with the RTI is performed. This module must comprise the standard methods required for the communication with the RTI. Some of this methods are used by the RTI to perform call-backs to the federate, others are the methods used by the federate to perform calls to the RTI. The last ones, uses a local

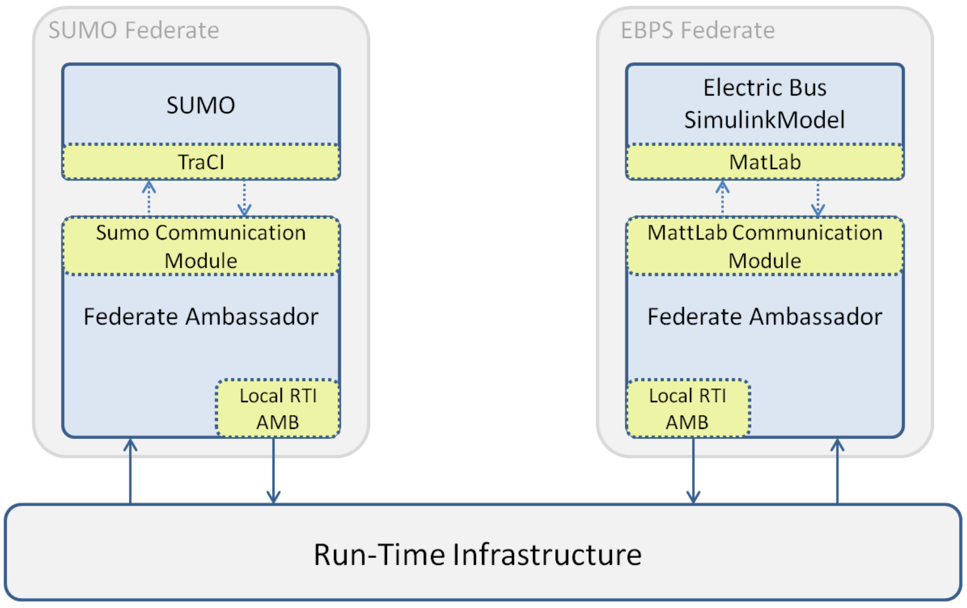


Figure 5.5: HLA Implementation Architecture

RTI component in order to perform the calls. The local RTI component is a local library (Which in this case, is a "jar" file) with the RTI methods.

5.3.2.1 Connect and Join Methods

In order to establish communication with the RTI the first thing that the federate must do is to perform a connection to it. To do that, it is used an object called RTI ambassador that is created using an *RTIambassadorFactory* provided by the local RTI library. This object will be used to perform all the calls to the RTI.

After the RTI ambassador object been created, the federate needs to connect to the RTI, and then join or create a specific federation as shown in Listing 5.4.

```
//Get the object of RTI Ambassador
RtiFactory rtiFactory = RtiFactoryFactory.getRtiFactory();
RTIambassador  = _ambassador = rtiFactory.getRtiAmbassador();

//Connect to the RTI
_ambassador.connect(federateAmbassador, IMMEDIATE, "MySettingsDesignator");

//Create the federation
_ambassador.createFederationExecution("MyFederation", "MyFOM");

```

```
//Join the federation
_ambassador.joinFederationExecution("MyName", "MyFederateType", "MyFederation");
```

Listing 5.4: Pseudo-code for RTI and federation connection

This methods as well as *resignFederationExecution*, *destroyFederationExecution* and *disconnect*, comprises the main activity for connection and disconnection from a federation. Now, the methods for the data exchange within federation need to be implemented. The HLA standards provide an extensive number of methods for different purposes. However, implementing them all would be very time consuming. Thus, only the necessary methods for the Integration intended were developed.

5.3.2.2 Data Exchange Methods

There are two different groups of methods that are related with the type of data exchange. The first group is directed to the interaction classes and the other, to the object classes.

Related to the interaction classes, a set of methods were used in order to allow one of federates to interact with the other. The first methods that needed to be used are the *PublishInteractionClass* and *SubscribeInteractionClass*. These methods acknowledge what kind of interaction the federates are able to publish or receive. For example, if a federate will be responsible to instruct the simulation to starts, this federate could publish an interaction class named "start" while the others subscribe it. In this case, the SUMO federate will be the responsible to initiate the simulation and though, to publish an interaction class named "Start". The EBPS federate will need to subscribe that interaction in order to receive it.

The others methods required for the interaction exchange are the *ReceiveInteraction* and *SendInteraction*. The SUMO federate needs to call the method *SendInteration* from RTI, whenever it wants to start the simulation. The EBPS federate needs to implement the *ReceiveInteraction* method in order to receive a callback from the RTI with the interaction and starts the simulation.

In a similar way each federate needs to call "Publish" or "Subscribe" for the *ObjectClass* and *ObjectClassAttributes* that they want to send or receive. Both, SUMO and EBPS federates needs to perform "Publish" and "Subscribe" to the Bus class. However, the EBPS only subscribes the attribute velocity and publish all the others. The SUMO federate performs the publish and subscribe attributes of the Bus class in the opposite way. Table 5.5 presents the *InteractionClasses*, *ObjectClasses* and *ObjectClasseAttributes* Published and Subscribed for each federate.

Now that the federates informed the RTI of what they Publish and Subscribe, they had to implement the necessary methods for exchanging data during federation execution. These methods are the following:

- registerObjectInstance
- discoverObjectInstance
- updateAttributeValues

Class Type	**Identifier**	**SUMO**	**EPBS**
InteractionClass	Start	P	S
ObjectClass	Bus	PS	PS
ObjectClassAttribute	Velocity	P	S
ObjectClassAttribute	Acceleration	S	P
ObjectClassAttribute	Power	S	P
ObjectClassAttribute	Torque	S	P
ObjectClassAttribute	TotalCycleEnergy	S	P
ObjectClassAttribute	BrakingKinectEnergy	S	P
ObjectClassAttribute	BrakingResistanceEnergy	S	P
ObjectClassAttribute	SuperCapacitorsChargingEnergy	S	P
ObjectClassAttribute	SuperCapacitorsDischargingEnergy	S	P
ObjectClassAttribute	BatteriesChargingEnergy	S	P

Table 5.5: Public and subscribe entities by SUMO and EBPS federates

- reflectAttributeValues
- attributeOwnershipAcquisition

The *registerObjectInstance* service registers a new object instance of the specified type. It returns a handle to the new instance. This handle is saved so that federate could update this object later on. In this specific case, the only federate that uses this service is the SUMO federate, being responsible for registering the class Bus in the federation. When a new object is registered by one federate, the RTI will make sure that it is discovered by other federates that subscribe to the specified class. This is made by the *discoverObjectInstance* method. If a federate joins a federation where there are already a number of objects, the RTI will also make sure the new federate discovers existing instances of a class that it subscribes to.

The *updateAttributeValues* service sends an attribute update for a particular object instance. The update contains a number of attribute/value pairs where the attribute is described by its handle. There are two cases to consider when the federation send updates for attributes: one is when a federate requests a value, the other is when a federate have a new value to update. In this implantation the only way to updates happens is when a federate receives a new value. The *reflectAttributeValues* is called-back by the RTI to the federate that subscribes the attributes that was updated by the other federate.

Finally, the *attributeOwnershipAcquisition* is used by the EBPS federate to request ownership for some attributes. Since it was the SUMO federate to register the class Bus, by default all the attributes were owned by it and the EBPS could not update any. Calling the *attributeOwnershipAcquisition* service to RTI, it can request the attributes that it want to update and gain their ownership.

With all methods implemented, their execution sequence will be performed as illustrated in Figure 5.6.

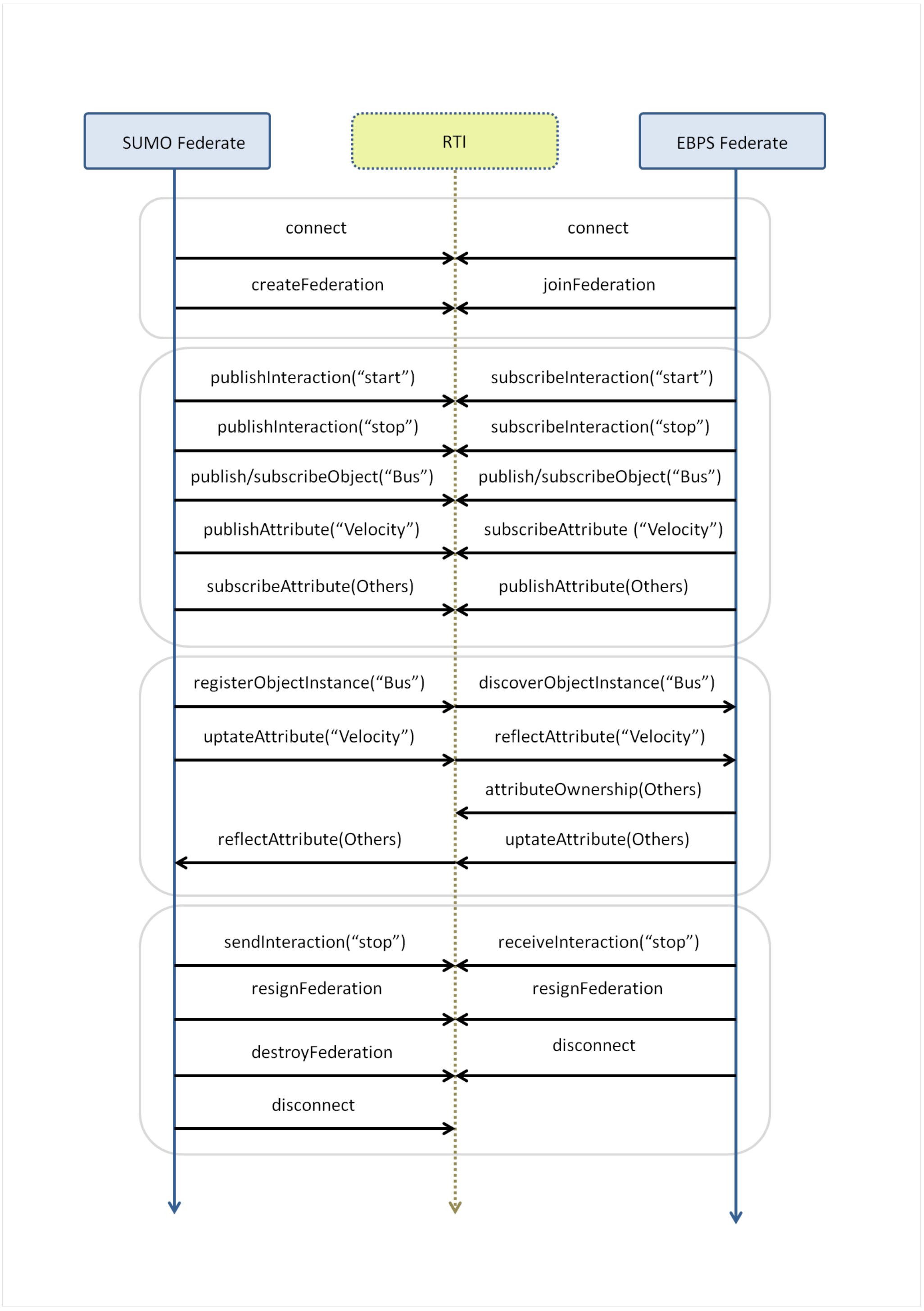

SUMO Federate
RTI
EBPS Federate
connect
connect
createFederation
joinFederation
publishInteraction(“start”)
subscribeInteraction(“start”)
publishInteraction(“stop”)
subscribeInteraction(“stop”)
publish/subscribeObject(“Bus”)
publish/subscribeObject(“Bus”)
publishAttribute(“Velocity”)
subscribeAttribute (“Velocity”)
subscribeAttribute(Others)
publishAttribute(Others)
registerObjectInstance(“Bus”)
discoverObjectInstance(“Bus”)
uptateAttribute(“Velocity”)
reflectAttribute(“Velocity”)
attributeOwnership(Others)
reflectAttribute(Others)
uptateAttribute(Others)
sendInteraction(“stop”)
receiveInteraction(“stop”)
resignFederation
resignFederation
destroyFederation
disconnect
disconnect

Figure 5.6: Diagram flux of federation execution

Chapter 6

Preliminary Results and Discussion

For a proper validation of the devised integration architecture and implementation, some measures have to be analysed to evaluate its effectiveness. In this chapter are presented functional and performance tests as well as the metrics that have been implemented to assess the system. A test-bed was developed in order to evaluate the system's behaviour and to demonstrate the usability of the integrated simulation in different fields of study.

6.1 Functional Tests

The features implemented in this project need validation to verify that the integration of the two simulation systems is indeed successfully accomplished. Thus, it has been performed a series of functional tests to identify possible issues. These tests are described bellow.

6.1.1 Connect both federates to the RTI, create a federation and join them to it

For the evaluation of correct connection of the federates to RTI it is necessary to run the federates and to verify that they connect to the RTI middle-ware and that the federation is indeed created. Thus, the following steps are taken to run the first test:

1. Start the pitch pRTI.
2. Start SUMO and EBPS
3. Run SUMO and then EBPS federates
4. Verify, through the RTI GUI, if the federates connect to the RTI and join to a federation

As it can be seen in Figure 6.1, both federates appear to be connected to the RTI and joined into the federation called "Electric Bus in Traffic Simulation". This way this test was successfully passed.

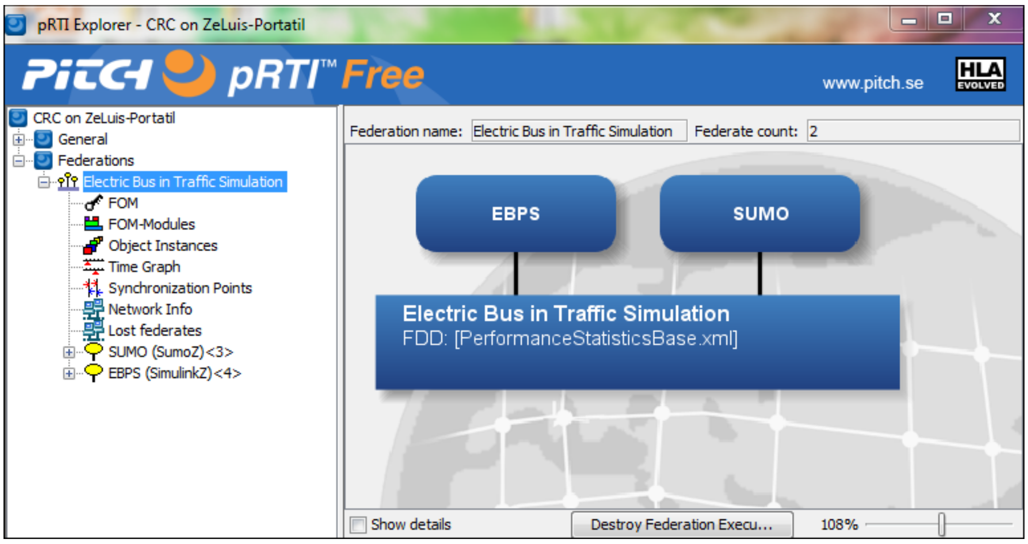


Figure 6.1: Pitch pRTI GUI with connected federates

6.1.2 Perform an interaction between federates

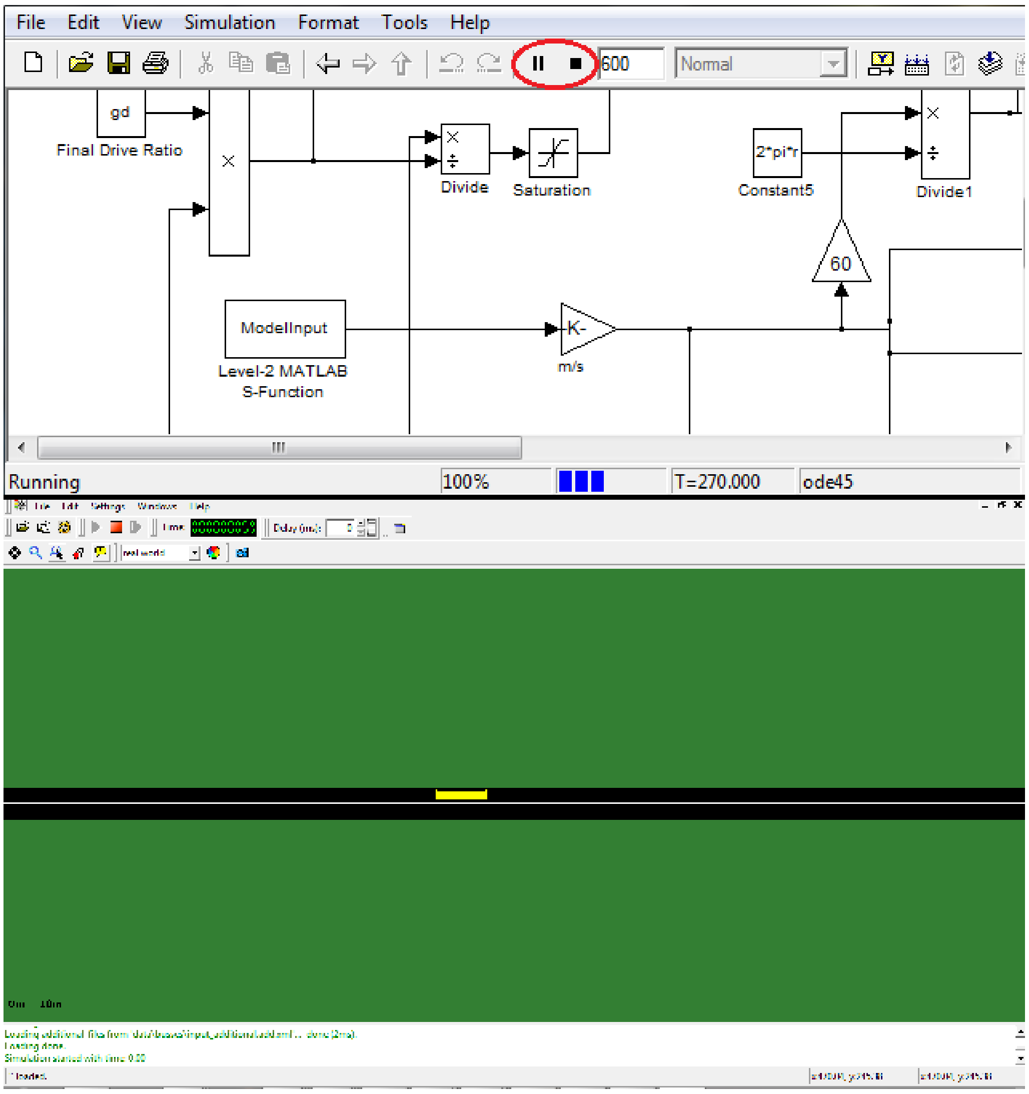
The following functional test is for verifying the interaction "Start". For this test it is added to SUMO federate the possibility to receive an input through the console. After have the federates joined the federation, the simulation will not start until SUMO federate receives the input and send the interaction "Start" to the EBPS federate. When the key "s" is pressed in the SUMO federate console, the interaction should be send and the simulation should start on both federates. The steps to perform this test are as follows:

1. Start the pitch pRTI.
2. Start SUMO federate loaded with a sample network (see Section 5.2.2);
3. Start EBPS federate
4. Press "s" in SUMO federate console
5. Verify, through the SUMO GUI and EBPS model GUI, if the federates start the simulation

As it can be seen in Figure 6.2 both federates appear connected to the RTI joined to the federation "Electric Bus in Traffic Simulation". This way this test was successfully completed.

6.1.3 Exchange data between federates

One of the most important features of the framework is the data exchange between the models. Thus, a functional test serves to validate the data exchange mechanism. Here, both federates will print the data send and the data received in their respective consoles. This way it will be possible to verify if the information arrived to its destiny. The steps to perform are the following:

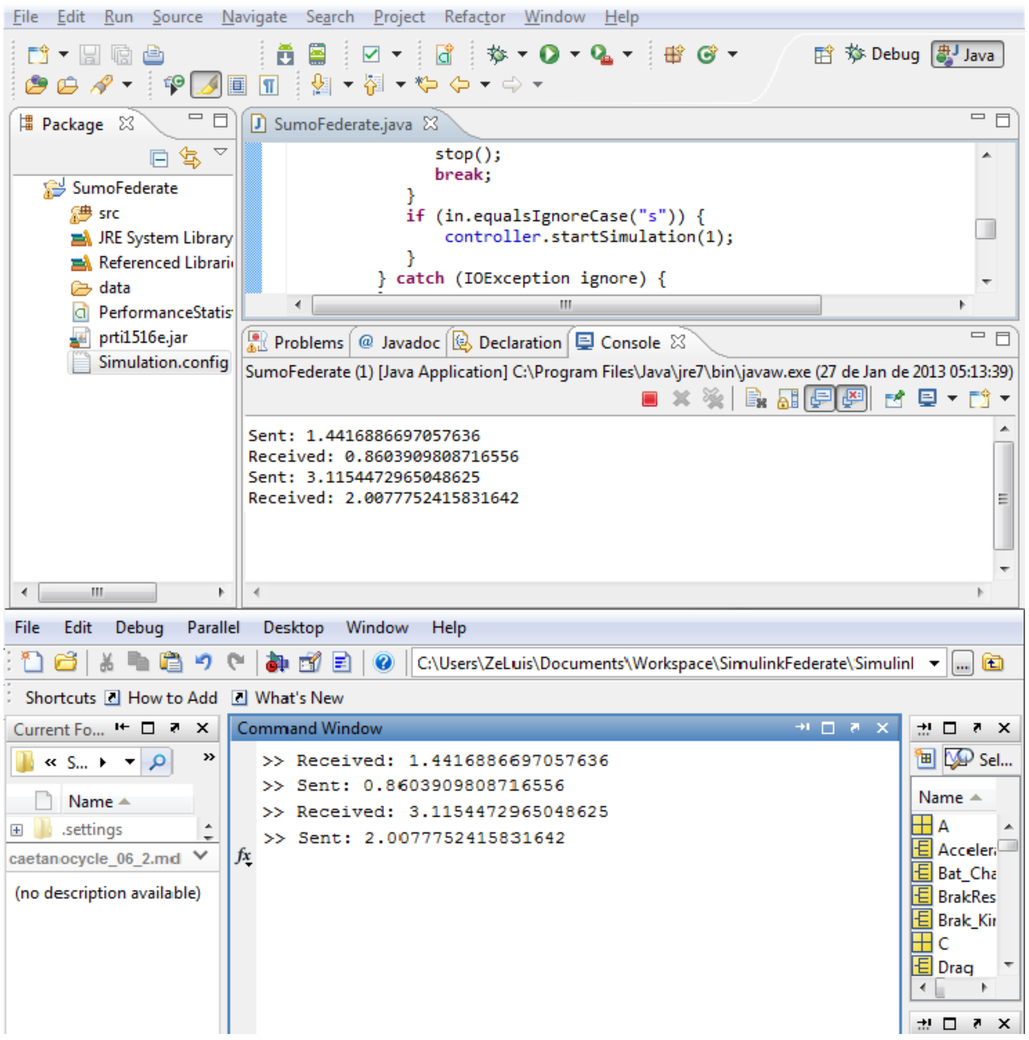


```
Loading additional files from 'data/...input_additional.add.xml' ... done (2ms).
Loading done.
Simulation started with time: 0.00
```

Figure 6.2: Simulation execution in both simulation tools

1. Start the pitch pRTI.
2. Start SUMO federate loaded with a sample network (see Section 5.2.2);
3. Start EBPS federate
4. Press "s" in SUMO federate console
5. Verify, through the SUMO and Matlab consoles, the sent and received information

As it can be seen in Figure 6.3, the transmitted information by one federate is the same with the information the other federate receives. This means that the test is successfully completed and the data exchange between federates within RTI is validated.



```
                stop();
                break;
            }
            if (in.equalsIgnoreCase("s")) {
                controller.startSimulation(1);
            }
        } catch (IOException ignore) {
```

SumoFederate (1) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (27 de Jan de 2013 05:13:39)

```
Sent: 1.4416886697057636
Received: 0.8603909808716556
Sent: 3.1154472965048625
Received: 2.0077752415831642
```

C:\Users\ZeLuis\Documents\Workspace\SimulinkFederate\Simulinl

```
>> Received: 1.4416886697057636
>> Sent: 0.8603909808716556
>> Received: 3.1154472965048625
>> Sent: 2.0077752415831642
```

Figure 6.3: Data information exchanged between federates

6.1.4 Validate integrated simulation results

After the validation of all the features the last test is related to the consistency of the integration. This test aims to validate the veracity of the exchanged data. It's exactly the same test performed for data validation within TrasMAPI integration (see Section 5.2.1). To perform this test, some changes has been made to SUMO federate in order to set the velocity of the car with the values of the field experimental data. In this way, the EBPS outputs should be validated with the outputs from the field experiment data. To run the test the following steps are taken:

1. Start the pitch pRTI.
2. Start SUMO federate loaded with a sample network (see Section 5.2.2);
3. Start EBPS federate
4. Press "s" in SUMO federate console
5. Verify the outputs sent from EBPS and compare with the field experiment data

From the obtained results the success of the test can be agreed. The results were exactly the same as the values from the data of the field experiment(see Appendix B).

6.2 Experimental Set-up

It has been demonstrated by the tests that the integration of both systems using HLA standards is valid and consistent. However, it is not demonstrated yet the potential of this framework for the intended purpose it was designed. In this section, a test-bed is presented to demonstrate the advantages of the distributed simulation for different field studies.

For this purpose a new road network has been designed and created for SUMO. The network is a model of the central area of Porto's down-town(Aliados) and it has been extracted from OpenStreetMaps database. Then, the model has been edited using the JOSM editor and by hand.

The choice of this urban traffic area is motivated by the fact that it is an area of the centre of the city with high traffic densities and a large number of traffic lights. Figure 6.4 illustrates the model network where in the zoomed part it can be distinguished a bus stop represented by a green rectangle. Although it is a simple network, it still serves our interests as it represents the centre of cities, where the network's topology becomes very dense.

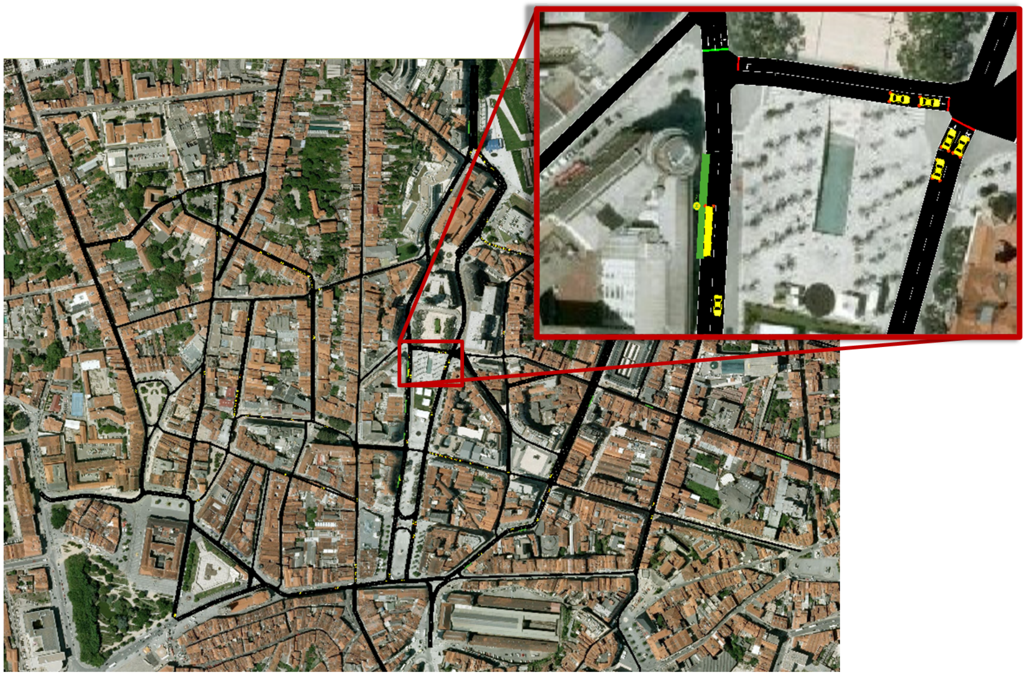


Figure 6.4: Aliados network for test-bed experiments.

For this test-bed two different case studies have been performed. The first scenario, was aiming to analyse the EBPS behaviour within different volumes of traffic. The second one had the goal to analyse the influence of the driver's behaviour in the EBPS performance. On both cases, the bus has followed the same route through Aliados avenue. The chosen metrics for the EBPS performance analysis are *acceleration*, *power*, *totalEnergyCycle*, *brakingKinectEnergy*, *superCapacitorsChargingEnergy* and *batteries*ChargingEnergy, already described in detail on section 4. In the first case, the acceleration parameter has not been comprised in the results since that is the same driver for the two scenarios and thus, the same acceleration behaviour. Furthermore, the outputs obtained during the simulation are compared and the results are properly explained and discussed.

Different Traffic Flows Analyses

For the first analysis, two different traffic flows have been set-up, so that the EBPS could be exposed to different solicitations due to daily traffic conditions. As it has been explained above, the trips are generated automatically by the *randomTrips* application embedded to SUMO package. So the only way to customize the traffic flow is to modify the repetition rate of the generated trips. For the first set-up a repetition of five seconds has been used for the traffic flow generation. Since one second represents one step in simulation, this means that each trip is repeated by a new vehicle at every step. The second set-up of traffic flow has been generated with a repetition trip rate of one second. With this rate, rapidly the network begins to become overloaded, representing rush hours periods.

In each scenario, the bus will travel through the same route. In this way, it will be possible to compare the results of the EBPS for the intended cases.

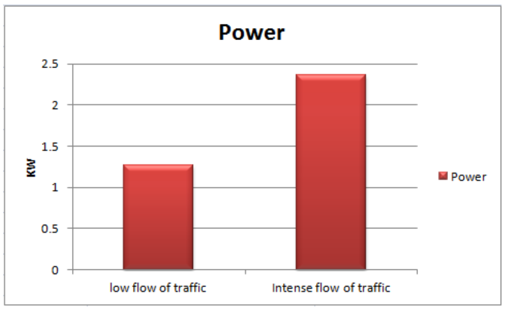


Figure 6.5: Total Power average (in KW)

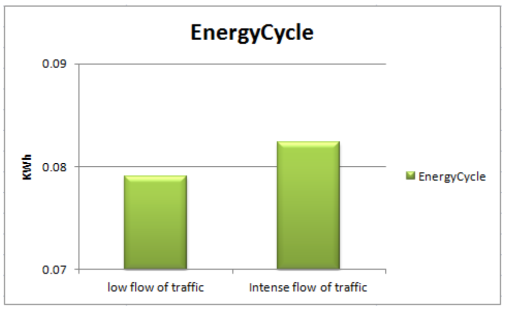


Figure 6.6: Necessary Energy to perform the trip (in KWh)

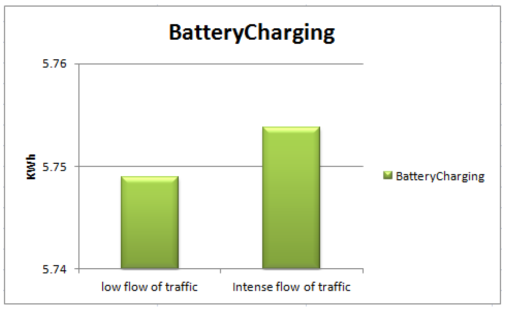


Figure 6.7: Total Battery Charging energy during the trip (in KWh)

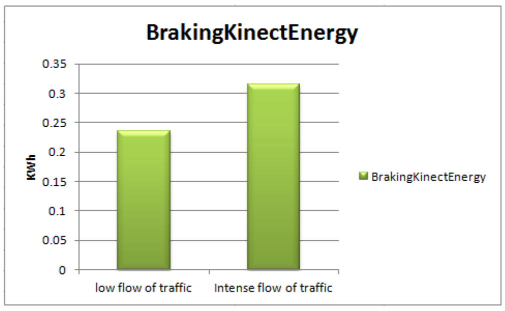


Figure 6.8: Total Braking Kinect Energy dissipated during the trip (in KWh)

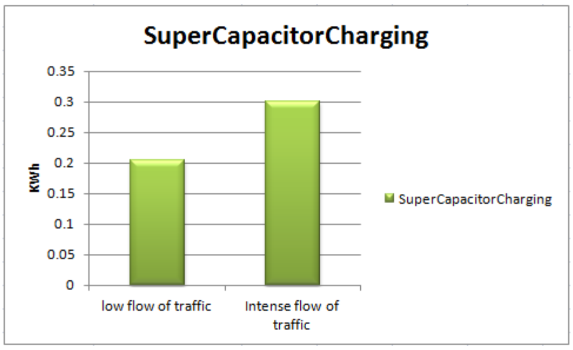


Figure 6.9: Total Super Capacitor Charging Energy during the trip (in KWh)

Figure 6.5 plots the average power used by the bus at each time-step. Figure 6.6 shows the necessary energy to perform the trip taking into account the energy that was recovered (i.e. if the trip consumed 5.83 KWh and 5.75 KWh was recovered, then the energy cycle is 0.08 KWh). Furthermore, Figure 6.7 illustrates the energy recovered by the battery from the engine rotations. Finally, the Figures 6.8 and 6.9 demonstrate the energy dissipated from braking episodes and the amount of that energy that is recovered by the super-capacitor respectively.

As it can be seen in Figure 6.5, the average power used by the bus at each time-step is significantly superior in the intense traffic flow scenario. However, the energy necessary to perform the trip is almost the same for both cases, as illustrated in Figure 6.6. Since the battery charging was also almost the same for the both scenarios, this could be explained by the fact that the energy dissipated in braking was higher on the intense traffic flow scenario and more quantity of energy were recycled by the super capacitors as can been seen in Figures 6.8 and 6.9.

This is one example of integrated studies that could be performed with the developed framework. For example, with these results, one may conclude that for this specific route, the performance of the electric bus will not suffer great impact with the increase of traffic volume.

Different Driver Behaviour Experiment

The second test serves to demonstrate the influence of the driver's behaviour on the EBPS performance. Thereby, two different types of driver have been developed: one with an aggressive driving style and another with a more tenuous one. The difference between this two types of driver is only in terms of acceleration and deceleration. The aggressive driver tries to reproduce the type of impatient driver who makes hard acceleration and stops too much on top of the situations. On the other hand, the tenuous driver tries to reproduce the kind of driver that performs soft accelerations and also starts to brake with some advance providing soft decelerations.

For the implementation of the two types of drivers, a second vehicle has been created in the SUMO's file having different values in the acceleration and deceleration variables. In this sense, the SUMO itself is responsible to reproduce the different behaviour of acceleration for both vehicles. Listing 6.1 illustrates the file contents for the description of the two different buses.

```
<vType id="BUS1" accel="2.6" decel="4.5" sigma="0.5" length="12" minGap="3"
    maxSpeed="70" color="1,1,0" guiShape="bus/city"/>
<vType id="BUS2" accel="5.2" decel="6.5" sigma="0.5" length="12" minGap="3"
    maxSpeed="70" color="1,1,0" guiShape="bus/city"/>
```

Listing 6.1: Description of the two different buses in SUMO's file

The *accel* and *decel* labels correspond to the acceleration and deceleration of the vehicle respectively. As one can see, the values of *accel* for the vehicle named "BUS2" are twice the values for vehicle named "BUS1" while the *decel* for "BUS2" is just two unites more than "BUS1". One must note that these values do not exceed the maximum acceleration and deceleration behaviours that a real bus can perform. These maximum values constraints have been provided along with the field data experiment results.

Just to conclude, this test has been performed within the first set of traffic flow density presented above, since it could highlight better the influence of the different drivers behaviours on EBPS performance.

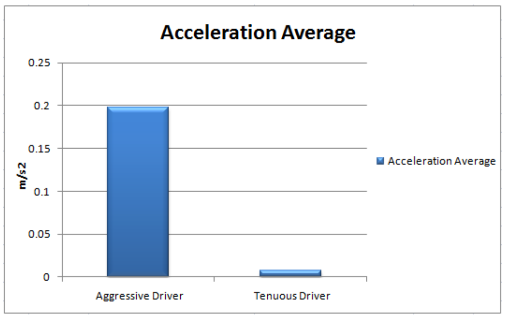


Figure 6.10: Total Acceleration average (in m/s^2)

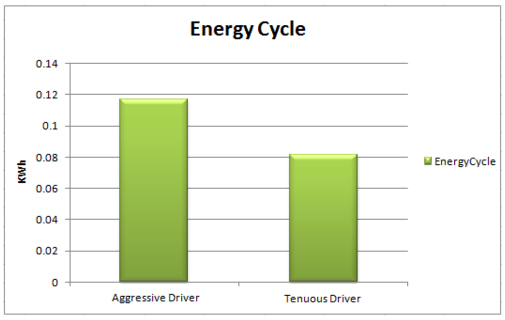


Figure 6.11: Necessary Energy to perform the trip (in KWh)

This case study aims to demonstrate that the driver behaviour could influence the performance of the EBPS. Thus, a simple comparison between the acceleration average and the energy cycle of each driver is presented to corroborate that premise.

As could be seen in Figure 6.10 the driver with aggressive behaviour has an acceleration average superior to the tenuous one, which is reflected in the total energy consumed during the route. Thereby, the aggressive driver requires more energy then the tenuous one, to perform the same route.

6.3 Functionality Tests

To evaluate the platform performance to be eventually compared against other frameworks the CPU usage of the integrated simulation will be assessed.

These test runs use the bundled operating system profiling tools to measure the CPU usage metric. Thus, on Windows 7 these metric are accessed using the "Performance" tool on Administrative Tools > Performance.

The evaluation tests have been performed on a desktop computer with the specifications illustrated in Table 6.1.

Components	**PC**
CPU	Intel Core i5
Clock speed	3.20GHz
RAM size	4GB
Graphics Card	GeForce 310
Operating System	Windows 7

Table 6.1: Computer set-up used to evaluate the integrated platform

The test has been performed using the test-bed example with the intense traffic volume aforementioned and evaluating the overall CPU usage of the all applications directly related to the simulation (i.e. Sumo, MatLab, Eclipse and Pitch pRTI) during the simulation. The achieved results from the test run is presented in Figure 6.12.

From the results, it is possible to verify that the CPU usage average is around 30% and that the maximum usage do not overcome the 40%. It looks that the framework works quite well for one electric bus on its network. However, this performance test should be performed with various buses to evaluate if this framework can hold coordination and data exchange without problems.

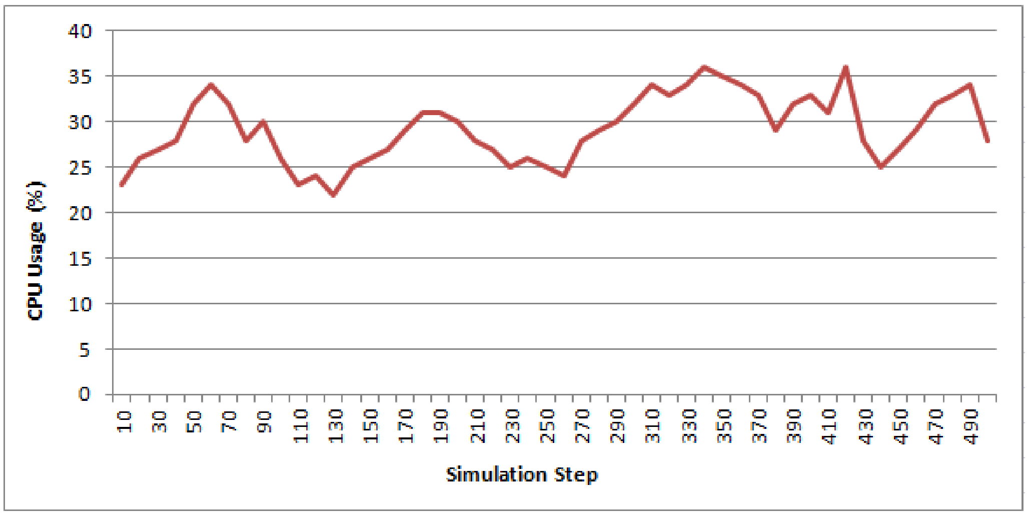


Figure 6.12: Overall Simulation performance with one electric bus

One could use the same EBPS federate to instantiate difference instances of MatLab an execute different simulations for the same model. For this approach some modifications are necessary to be performed at the federate ambassador level. However one should account with possible bottleneck in this federate during the control of all EBPS Simulink models.

6.4 Summary

This chapter has presented some simple functional tests with the purpose to evaluate the implemented integration platform feasibility. From the results of functional tests it can be acknowledged the successful development of the HLA integration framework and so the implementation of all its components.

The performance tests have demonstrated that with a relative recent desktop computers with the specifications similar to the ones aforementioned in Figure 6.1, the framework performance behaves quite well. However, one must note that these tests have been performed just using one electric bus due the fact that the RTI only allows two federates. Furthermore, two case studies were presented to demonstrate the usefulness of the proposed framework, for example planning routes for the electric buses according to their performance in such areas, and studies on the influence of drivers behaviour on the electric bus performance.

The next chapter will conclude this work with a general overview of every chapters discussed, the main achieved results and a future perspective onto further developments of the platform.

Chapter 7

Conclusions and Future Work

The growth of traffic and transport demand observed over the past years is closely related to the principal difficulties we struggle to solve today. Traffic congestion influence not only the economic activity of cities but is also responsible for air quality and global warming problems.

It is true that, gas emissions from vehicles represent one of the major causes of the greenhouse effect. For this reason (also) institutions are constantly looking for different alternatives especially in public transportations. In this sense, incentives and investments on public transports as well as research on more eco-sustainable solutions have been performed as attempts to minimize both the air pollution and congestion problems. One of the approaches currently investigated and implemented to provide an eco-sustainable solution is related to the employment of electric bus powertrains in metropolitan transportation as an alternative to internal combustion engine buses.

However there are still open issues related to the consumption of energy and other performance measures for considering the adoption of electric buses in urban scenarios as a cost-effective solution.

Traffic simulation already implements tools to analyse traffic networks layouts and traffic control strategies. Nevertheless, few of these tools provide the possibility of the integration of electric vehicle subsystems with a traffic domain models. Indeed, an important aspect in evaluating the performance and adequateness of such vehicles is the fact they are to be in immersed into an urban environment context.

It is a general true that a route having many positive elevations or a traffic congestion situation will directly affect the autonomy and performance of the vehicle. The goal of this thesis is bridge the gap between urban mobility analysis and testing electric bus' autonomy and performance. Thus, it has been presented a distributed architecture for electric bus powertrain simulation within a realistic urban mobility context. Such a platform will be important for analysing how traffic flow and its dynamics affect the performance of the electric bus when there are obstructions or intense traffic conditions.

In the following sections, a critical analysis of the implemented solution overlooking the original objectives for this dissertation will be outlined, along with its main results. Last, some developments to improve the platform and its usability are suggested.

7.1 Overview

As stated throughout this document, the main objective of this project was to verify the possibility of coupling two simulators, a traffic simulator and an electric bus subsystem model, to widen the possibility of studies of the EBPS within a common traffic environment.

In the literature review chapter, the areas of knowledge that this project involves have been explored and the current state of the art has been presented. It has been underlined the importance of combining different aspect of the "real system" especially in a complex domain such traffic. So, integrations of microscopic and macroscopic models integrations demonstrated to be helpful for observing the overall performance in large-scale simulation controlling the micro aspects in specific areas of interest.

Thus, the computational effort for computing large networks could be minimized. Moreover, integrations with micro a nanoscopic models demonstrated advantages in testing stand-alone models for example autonomous vehicles, in microscopic traffic simulation environment. Some projects were already developed like embedded models in SUMO's core, but these approaches are not flexible and do not comprise all of the vehicle's characteristics, such as the performance metrics.

Furthermore, in the methodological approach chapter, an integrated architecture has been planned, pointing the main aspects towards an efficient communication among simulators. This flexible approach has been devised under the HLA (High Level Architecture) concepts, thus opening new opportunities to the integration with different simulation platforms.

A prototype using both SUMO and EBPS has been also devised accordingly to the integration requirements. First, a description of the chosen software has been presented, explaining the main features and their relevance on the core of the project. The main development steps have been thoroughly described. To couple these two simulators, has been followed an integration using TraSMAPI layer as a first approach, and then an integration under HLA concepts using the Pitch pRTI has been performed. The advantages of using the HLA standards derive from the necessity to obtain interoperability and reuse legacy models. The traffic simulator provided a bus velocity to the EBPS one, whereas the latter calculates the performance metrics, feeding back their values to the traffic simulator in the same time step.

For test and validate the devised framework, it has been modelled and developed a scenario that comprehends a bus navigation through a simple route of the Porto's city. First, functionality tests were performed to validate the integrated framework, then, to evaluate the effectiveness of the framework, performance tests were performed.

7.2 Main Contributions

Considering the framework implementation outcome and its stated performance measures, it can be agreed that the project has been concluded quite successfully.

The proposed architecture envisioned for a flexible approach to the integration of two simulators, one from the transportation area, and the other from automotive area. The first accomplished objective is the development of a communication module allowing total control of a Simulink model simulation so it could be externally controlled. This communication module can now be used in order to expeditiously interact with different Simulink models. Other accomplished objective, similar to previous one, is the communication with SUMO through TraCI, allowing complete control of simulation and access to vehicles attributes.

The integration phase of the project consists on two different approaches, one using TraSMAPI and other using an RTI. Regarding TraSMAPI integration, an adaptation of its framework has been performed in order to allow connection with two different simulators at a time. TraSMAPI aims to allow real-time communication with microscopic simulators providing a framework for the development of multi-agent solutions. With these adaptations, now it can allow the development of multi-agent solutions to be tested in different simulators and at the same time to evaluate and compare the different results. Also, it can allow multi-agent solutions to be used in different paradigms of simulation, using different simulation models and allowing data exchanges between them.

In order to integrate the two simulators under HLA concepts, the specification of both SUMO and EBPS SOMs (Simulation Object Models) have been devised as well as both federate ambassadors. With the developed assets, one can use any of this simulation tools for other simulation purpose under HLA, without great efforts.

For testing and evaluating the integrated framework a simple network of Aliados has been modelled, which provided a fairly realistic environment for the EBPS. Also, the implemented test-bed demonstrated the ease of use the platform and some possibles studies that could made with it.

7.3 Future Work

Apart from the initial goals defined for this work, there is room for further improvement as new features and capabilities have been envisaged during its development.

The studies of the EPBS simulator could greatly benefit from the addition of more complex driver behaviours with intelligent agents. In particular, consider the possibility of agents having some form of proactive behaviour and adaptivity to the outcomes from the EBPS.

The simulation would equally be improved with the inclusion of a third dimension in the topologies of traffic roads, which would lead to more realistic and efficient evaluations of the EBPS performance. Currently, a new version of the EBPS which already considers the elevation on its performance is under validation. Thus, this new version of the model could be used alongside with other simulator that provides the third dimension parameter, or modify the SUMO simulator so that it could comprise that information.

To perform more realistic case studies, real bus routes could be modelled with realistic approximations of the stop times in each bus stop. To do so, probably a larger Porto's network would be modelled too.

Another future development is the inclusion of another simulator in the federation that could bring some value to the electric bus analysis, for example a robotics simulator for autonomous driving. In this sense, an autonomous electric bus could be studied in urban traffic situations. With three federates cooperating among each other, the coordination and data exchange capabilities of the RTI would be deeper studied and tested, as well as some methods, for example, for exchanging attributes ownership.

References

[AOC] Martin Adelantado, Armand Oyzel, and Jean-baptiste Chaudron. Using the HLA , Physical Modeling and Google Earth for Simulating Air Transport Systems Environmental Impact.

[ARS07] R Arnott, T Rave, and R Schöb. Alleviating urban traffic congestion. *MIT Press Books*, 1, 2007.

[BACT06] M E Ben-Akiva, C Choudhury, and T Toledo. Lane changing models. In *Proceedings of the International Symposium of Transport Simulation*, 2006.

[BBEK11] M Behrisch, L Bieker, J Erdmann, and D Krajzewicz. SUMO - Simulation of Urban MObility: An Overview. In *SIMUL 2011, The Third International Conference on Advances in System Simulation*, pages 63–68, Barcelona, Spain, 2011.

[Bra68] D Braess. Über ein Paradoxon aus der Verkehrsplanung. *Mathematical Methods of Operations Research*, 12(1):258–268, 1968.

[Bur04] Wilco Burghout. Hybrid microscopic-mesoscopic traffic simulation. *Transportation Research Record: Journal of the Transportation Research Board*, (Volume 1934 / 2005):218–255, 2004.

[Cal] Caliper. TransModeler Traffic Simulation Software. Avaiable in `http://www.caliper.com/transmodeler/default.htm`, accessed in 20 December 2012.

[Car04] John S Carson II. Introduction to modeling and simulation. In *Proceedings of the 36th conference on Winter simulation*, WSC '04, pages 9–16. Winter Simulation Conference, 2004.

[CH09] Rutger Claes and Tom Holvoet. Multi-model traffic microsimulations. *Proceedings of the 2009 Winter Simulation Conference (WSC)*, pages 1113–1123, December 2009.

[CPD$^+$00] C G Cassandras, C G Panayiotou, G Diehl, W-b Gong, Z Liu, and C Zou. Clustering Methods for Multi-Resolution Simulation Modeling. *Proc. Conf. Enabl. Technol. Simul. Sci., Int. Soc. Opt. Eng., Orlando, USA*, pages 37–48, 2000.

[Dag97] Carlos. Daganzo. *Fundamentals of transportation and traffic operations*. Pergamon, Oxford; New York, 1997.

[DH01] J B Dabney and T L Harman. *Mastering Simulink 4*. Prentice Hall PTR, 2001.

[Dow04] A Downs. *Still stuck in traffic: coping with peak-hour traffic congestion*. Brookings Inst Press, 2004.

[DP08] H Dia and S Panwai. Nanoscopic traffic simulation: enhanced models of driver behaviour for ITS and telematics simulations. In *International Symposium on Transport Simulation, 8th, 2008, Surfers Paradise, Queensland, Australia*, 2008.

[DPW10] M Doering, T Pögel, and L Wolf. DTN routing in urban public transport systems. In *Proceedings of the 5th ACM workshop on Challenged networks*, pages 55–62. ACM, 2010.

[DRE02] D T Drewry, Jr. Reynolds P.F., and W R Emanuel. An optimization-based multi-resolution simulation methodology. In *Simulation Conference, 2002. Proceedings of the Winter*, volume 1, pages 467 – 475 vol.1, 2002.

[DSN08] D Djenouri, W Soualhi, and E Nekka. VANET's mobility models and overtaking: an overview. In *Information and Communication Technologies: From Theory to Applications, 2008. ICTTA 2008. 3rd International Conference on*, pages 1–6. IEEE, 2008.

[FCD12] S Faye, C Chaudet, and I Demeure. A distributed algorithm for multiple intersections adaptive traffic lights control using a wireless sensor networks. In *Proceedings of the first workshop on Urban networking*, pages 13–18. ACM, 2012.

[FRBR09] M C Figueiredo, R Rossetti, R Braga, and L P Reis. An approach to simulate autonomous vehicles in urban traffic scenarios. In *Intelligent Transportation Systems, 2009. ITSC '09. 12th International IEEE Conference on*, pages 1–6, 2009.

[Fuj01] R M Fujimoto. Parallel simulation: parallel and distributed simulation systems. In *Proceedings of the 33nd conference on Winter simulation*, pages 147–157. IEEE Computer Society, 2001.

[GNAO12] J García-Nieto, E Alba, and A Carolina Olivera. Swarm intelligence for traffic light scheduling: Application to real urban areas. *Engineering Applications of Artificial Intelligence*, 25(2):274–283, 2012.

[GRM97] N.H. Gartner, A.K. Rathi, and C. Messer. *Monograph on Traffic Flow Theory*. PhD thesis, 1997.

[GTL$^+$09] J. Gozálvez, S. Turksma, L. Lan, O. Lazaro, F. Cartolano, E. Robert, D. rajzewicz, R. Bauza, F. Filani, and M. Röckl. iTETRIS: the Framework for Large-Scale Research on the Impact of Cooperative Wireless Vehicular Communications Systems in Traffic Efficiency. *Information and Communications Technologies (ICT-MobileSubmit 2009)*, 2009.

[Gui98] M U Guide. The mathworks. *Inc., Natick, MA*, 5, 1998.

[IEE10a] IEEE Std 1, editor. *IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Federati Interface Specification*. IEEE Computer Society, 2010.

[IEE10b] IEEE std 2, editor. *IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Framework and Rules*. IEEE Computer Society, 2010.

[IEE10c] IEEE std 3, editor. *IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Object Model Template Specification*. IEEE Computer Society, 2010.

[IL02] L H Immers and S Logghe. Traffic flow theory. *Faculty of Engineering, Department of Civil Engineering, Section Traffic and Infrastructure, Kasteelpark Arenberg*, 40, 2002.

[KBM$^+$05] D Krajzewicz, E Brockfeld, J Mikat, J Ringel, C Rössel, W Tuchscheerer, P Wagner, and R Wösler. Simulation of modern traffic lights control systems using the open source traffic simulation SUMO. In *Proceedings of the 3rd industrial simulation conference*, volume 2205, 2005.

[KDW00] F Kuhl, J Dahmann, and R Weatherly. *Creating computer simulation systems: an introduction to the high level architecture*. Prentice Hall PTR, 2000.

[KHRW02] Daniel Krajzewicz, Georg Hertkorn, Christian Rössel, and Peter Wagner. SUMO (Simulation of Urban MObility); An open-source traffic simulation. In *4th Middle East Symposium on Simulation and Modelling (MESM2002)*, pages 183–187, Sharjah / United Arab Emirates, 2002. SCS European Publishing House.

[KHWR02] Daniel Krajzewicz, Georg Hertkorn, Peter Wagner, and Christian Rössel. An Example of Microscopic Car Models Validation using the open source Traffic Simulation SUMO Sumo-netconvert Sumo-router. *Proc. 4th European Simulation Symposium*, 2002.

[KSSM98] U Klein, T Schulze, S Strassburger, and H P Menzler. Distributed traffic simulation based on the high level architecture. In *Proceedings of the Simulation Interoperability Workshop*, 1998.

[LC08] K Lan and C M Chou. Realistic mobility models for vehicular ad hoc network (VANET) simulations. In *ITS Telecommunications, 2008. ITST 2008. 8th International Conference on*, pages 362–366. IEEE, 2008.

[LK91] A. M. Law and W. D. Kelton. *Simulation Modeling and Analysis*. McGraw-Hill, New York, 3rd editio edition, 1991.

[LMJR04] H X Liu, W Ma, R Jayakrishnan, and W Recker. Large-Scale Traffi c Simulation Through Distributed Computing of Paramics. 2004.

[LTV$^+$06] P Laborczi, A Torok, L Vajda, S Kardos, G Gordos, and G Gerháth. Vehicle-to-vehicle traffic information system with cooperative route guidance. In *Proceeding of the 13th World Congress on Intelligent Transport Systems*, 2006.

[LW55] M.H. Lighthill and G.B. Whitham. On kinematic waves II: a theory of traffic flow on long crowded roads. *Proceedings of the Royal Society of London, series A,*, page 229, 1955.

[Mac] Christopher MacKechnie. Electric Buses - An Introduction. Avaiable in `http://publictransport.about.com/od/Transit_Vehicles/a/Electric-Buses-An-Introduction.htm`, accessed in 18 December 2012.

[Mar97] Anu Maria. Introduction to modeling and simulation. In *Proceedings of the 29th conference on Winter simulation*, WSC '97, pages 7–13, Washington, DC, USA, 1997. IEEE Computer Society.

[Mata] MathWorks. MATLAB the Language of Technical Computing. Avaiable in `http://www.mathworks.com/products/matlab/`, accessed in 13 January 2013.

[Matb] Matlabcontrol. matlabcontrol - A Java API to interact with MATLAB. Avaiable in `http://code.google.com/p/matlabcontrol/`, accessed in 1 January 2013.

[Mat99] MathWorks. Simulink - Dynamic System Simulation for MATLAB. *MathWorks, Inc.*, 1999.

[May90] Adolf D May. *Traffic flow fundamentals*. Prentice Hall, Englewood Cliffs, N.J., 1990.

[MCBB98] L Montero, E Codina, J Barceló, and P Barceló. Combining macroscopic and microscopic approaches for transportation planning and design of road networks. In *Proceedings of the 19 th ARRB Transport Research Conference, Sydney*, 1998.

[MM01] M D Meyer and E J Miller. *Urban transportation planning: A decision-oriented approach*. 2001.

[MML$^+$08] B Möller, K L Morse, M Lightner, R Little, and R Lutz. HLA Evolved–A Summary of Major Technical Improvements. In *Proceedings of 2008 Spring Simulation Interoperability Workshop, 08F-SIW-064*, 2008.

[MSAN11] R Maia, M Silva, R Araujo, and U Nunes. Electric vehicle simulator for energy consumption studies in electric mobility systems. In *Integrated and Sustainable Transportation System (FISTS), 2011 IEEE Forum on*, pages 227–232. IEEE, 2011.

[MSLZ11] Jian Ma, Jian Sun, Keping Li, and Liyan Zhang. A study on multi-resolution scheme of macroscopic-microscopic traffic simulation model. *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 1421–1426, October 2011.

[MSTR12] J Macedo, M Soares, I Timoteo, and R J F Rossetti. An approach to advisory-based traffic control. In *Information Systems and Technologies (CISTI), 2012 7th Iberian Conference on*, pages 1–6, 2012.

[Mur70] J D Murchland. Braess's paradox of traffic flow. *Transportation Research*, 4(4):391–394, 1970.

[Nel] Charlie Nelson. Deterministic and Stochastic Models. Avaiable in `http://www.futuretoolkit.com/detstoch.htm`, accessed in 08 January 2013.

[Ö09] T I Ören. Modeling and simulation: A comprehensive and integrative view. *Agent-Directed Simulation and Systems Engineering*, 78, 2009.

[OT04] J J Olstam and A Tapani. *Comparison of Car-following models*. Swedish National Road and Transport Research Institute, 2004.

[PR12] J L F Pereira and R J F Rossetti. An integrated architecture for autonomous vehicles simulation. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, pages 286–292. ACM, 2012.

[Pra09] C G Prato. Route choice modeling: past, present and future research directions. *Journal of Choice Modelling*, 2(1):65–100, 2009.

[PRRA12] Deborah Perrotta, Bernardo Ribeiro, Rosaldo J F Rossetti, and João L. Afonso. On the potential of regenerative braking of electric buses as a function of their itinerary. *Euro Working Group On Transportation*, 2012.

[Pur99] M Pursula. Simulation of traffic systems-an overview. *Journal of Geographic Information and Decision Analysis*, 3(1):1–8, 1999.

[Rob07] S Robinson. Conceptual modelling for simulation Part I: definition and requirements. *Journal of the Operational Research Society*, 59(3):278–290, 2007.

[Sar05] R G Sargent. Verification and validation of simulation models. In *Proceedings of the 37th conference on Winter simulation*, pages 130–143. Winter Simulation Conference, 2005.

[SC05] J Siegel and J E Coeymans. An integrated framework for traffic analysis combining macroscopic and microscopic models. *Transportation Planning and Technology*, 28(2):135–148, 2005.

[SD08] C Sommer and F Dressler. Progressing toward realistic mobility models in VANET simulations. *Communications Magazine, IEEE*, 46(11):132–137, 2008.

[SL09] David Schrank and Tim Lomax. The 2009 URBAN MOBILITY REPORT. Technical report, Texas Transportation Institute, The Texas A&M University System, 2009.

[SN93] M Simulink and M A Natick. The Mathworks. *Inc., Natick, MA*, 1993.

[Sov10] Benjamin K Sovacool. A transition to plug-in hybrid electric vehicles (PHEVs): why public health professionals must care. *Journal of epidemiology and community health*, 64(3):185–7, March 2010.

[ST70] Joseph William Schmidt and Robert Edward Taylor. *Simulation and Analysis of Industrial Systems*. Homewood, Illinois, 1970.

[SUM] SUMO:. G. Hertkorn, D. Krajzewicz, C. Rössel. 2002. *SUMO Homepage. http://sumo.sourceforge.net*.

[SWL11] Jason Sewall, David Wilkie, and Ming C Lin. Interactive Hybrid Simulation of Large-Scale Traffic. *ACM Transaction on Graphics (Proceedings of SIGGRAPH Asia)*, 30(6), 2011.

[SZ06] H. Shi and A. Ziliaskopoulos. A Hybrid Mesoscopic-Microscopic Traffic Simulation model: Design, Implementation and Compuatational Analysis. *85th Meeting of Transportation Research Board, CD-ROM.*, 2006.

[TARO10] I.J.P.M. Timoteo, M R Araujo, R J F Rossetti, and E C Oliveira. TraSMAPI: An API oriented towards Multi-Agent Systems real-time interaction with multiple Traffic Simulators. In *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*, pages 1183–1188, 2010.

[TARO12] I.J.P.M. Timóteo, M R Araújo, R J F Rossetti, and E C Oliveira. Using TraSMAPI for the assessment of multi-agent traffic management solutions. *Progress in Artificial Intelligence*, pages 1–8, 2012.

[Tra] Transport Simulation Systems. Aimsun. Avaiable in `http://www.aimsun.com/wp/`, accessed in 29 December 2012.

[Ty10] Aud Tennø y. Why we fail to reduce urban road traffic volumes: Does it matter how planners frame the problem? *Transport Policy*, 17(4):216–223, 2010.

[UBW$^+$10] N Unger, T C Bond, J S Wang, D M Koch, S Menon, D T Shindell, and S Bauer. Attribution of climate forcing to economic sectors. *Proceedings of the National Academy of Sciences*, 107(8):3382–3387, 2010.

[VO11] M Vasirani and S Ossowski. A computational market for distributed control of urban road traffic systems. *Intelligent Transportation Systems, IEEE Transactions on*, 12(2):313–321, 2011.

[Wha] What Is MATLAB. What Is MATLAB. Avaiable in `http://cimss.ssec.wisc.edu/wxwise/class/aos340/spr00/whatismatlab.htm`, accessed in 18 December 2012.

[WS06] Mohamed Wahba and Amer Shalaby. MILATRAS: A Microsimulation Platform for Testing Transit-ITS Policies and Technologies. *Proceedings of the IEEE Intelligent Transportation Systems Conference*, pages 1495–1500, 2006.

[WUW$^+$12] T Wongpiromsarn, T Uthaicharoenpong, Y Wang, E Frazzoli, and D Wang. Distributed Traffic Signal Control for Maximum Network Throughput. *arXiv preprint arXiv:1205.5938*, 2012.

[YCC10] Yali Yang, Hao Chen, and Lihua Chen. Evaluation of public transportation system in shanghai, china. *Computer and Communication Technologies in Agriculture Engineering (CCTAE)*, 2:197–199, 2010.

[Ye12] Sun Ye. Research on Urban Road Traffic Congestion Charging Based on Sustainable Development. *Physics Procedia*, 24, Part B(0):1567–1572, 2012.

[YLBO07] Levent Yilmaz, Alvin Lim, Simon Bowen, and Tuncer Oren. Requirements and design principles for multisimulation with multiresolution, multistage multimodels. *2007 Winter Simulation Conference*, pages 823–832, December 2007.

[YM06] Q. Yang and D. Morgan. A Hybrid Traffic Simulation Model. 85th Meeting of the Transportation Research Board. *85th Meeting of the Transportation Research Board CD-ROM, Washington DC.*, 2006.

[ZDHB09] S K Zegeye, B De Schutter, J Hellendoorn, and E A Breunesse. Integrated macroscopic traffic flow and emission model based on METANET and VT-micro. *Models and Technologies for Intelligent Transportation Systems*, pages 86–89, 2009.

[ZPK00] B P Zeigler, H Praehofer, and T G Kim. *Theory of modeling and simulation: Integrating discrete event and continuous complex dynamic systems*. Academic Pr, 2000.

[ZY04] X Zhang and H Yang. The optimal cordon-based network congestion pricing problem. *Transportation Research Part B: Methodological*, 38(6):517–537, 2004.

Appendix A

S-Functions Implementation

The S-Functions figuring in this appendix were used in the creation of a customized Simulink blocks, allowing external applications to access the input and output variables at each time-step of the simulation, through set_param() MatLab function.

A.1 S-Function ModelInput

```
function ModelInput(block)

%MSFUNTMPL A template for an M-file S-function

%   The M-file S-function is written as a MATLAB function with the

%   same name as the S-function.

%

%   It should be noted that the M-file S-function is very similar

%   to Level-2 C-Mex S-functions. You should be able to get more

%   information for each of the block methods by referring to the

%   documentation for C-Mex S-functions.

%

%   Copyright 2003-2006 The MathWorks, Inc.

%   $Revision: 1.1.6.13 $



```

```
%%

%% The setup method is used to setup the basic attributes of the

%% S-function such as ports, parameters, etc. Do not add any other

%% calls to the main body of the function.

%%

setup(block);



%endfunction



%% Function: setup ===================================================

%% Abstract:

%%   Set up the S-function block's basic characteristics such as:

%%   - Input ports

%%   - Output ports

%%   - Dialog parameters

%%   - Options

%%

%%   Required         : Yes

%%   C-Mex counterpart: mdlInitializeSizes

%%

function setup(block)



  % Register number of ports

  nu = 0; % 0 input ports

  ny = 1; % 1 output port
```

```



block.NumInputPorts  = nu;

block.NumOutputPorts = ny;



block.NumContStates = 0;



% Setup port properties to be inherited or dynamic

block.SetPreCompOutPortInfoToDynamic;



for i = 1:ny

 block.OutputPort(1).Dimensions = size(1);

 block.OutputPort(1).SamplingMode = 'sample';

 block.OutputPort(1).DatatypeID  = 0; % double

 block.OutputPort(1).Complexity  = 'Real';

 %block.OutputPort(1).DimensionsMode = 'Variable';

end



% Register parameters

block.NumDialogPrms     = 1;



block.DialogPrmsTunable = {'Nontunable'};



% Register sample times

%  [0 offset]            : Continuous sample time

```

```
 %  [positive_num offset] : Discrete sample time

 %

 %  [-1, 0]               : Inherited sample time

 %  [-2, 0]               : Variable sample time

 block.SampleTimes = [0 0];



 %% ----------------------------------------------------------------

 %% Options

 %% ----------------------------------------------------------------

 % Specify if Accelerator should use TLC or call back into

 % M-file

% block.SetAccelRunOnTLC(false);





 %% ----------------------------------------------------------------

 %% Register callback methods

 %% ----------------------------------------------------------------

 %block.RegBlockMethod('PostPropagationSetup',    @DoPostPropSetup);

 %block.RegBlockMethod('CheckParameters', @CheckPrms); % allow validation of block
     's dialog parameters

 %block.RegBlockMethod('ProcessParameters', @ProcessPrms); % Called in order to
     allow update of run-time parameters

 %block.RegBlockMethod('InitializeConditions', @InitializeConditions); % Called in
      order to initialize state and work area values

 %block.RegBlockMethod('Start', @Start);

 block.RegBlockMethod('Outputs', @Outputs); % Called to generate block outputs in
     simulation step
```

```
%% Matlab callbacks:

function DoPostPropSetup(block)

  %% Setup Dwork
  block.NumDworks = 1;

  %% [Slice maxSlice]
  block.Dwork(1).Name          = 'inputIndex';
  block.Dwork(1).Dimensions    = 1;
  block.Dwork(1).DatatypeID    = 7;
  block.Dwork(1).Complexity    = 'Real';

%endfunction

function CheckPrms(block)
%endfunction

function ProcessPrms(block)

%  block.AutoUpdateRuntimePrms;
%  block.OutputPort(opIdx).CurrentDimensions = 1;
```

```matlab




%endfunction

function InitializeConditions(block)

%  block.Dwork(1).Data = 1;



%endfunction



function Start(block)



  %% Initialize Dwork

%  block.Dwork(1).Data = 1;

%endfunction



function Outputs(block)

%   block.OutputPort(1).Data = evalin('base', block.DialogPrm(1).Data)

    block.OutputPort(1).Data = evalin('base', 'uWorkspace');

%   assignin('base', block.DialogPrm(1).Data, block.InputPort(1).Data(1));



%endfunction





function Terminate(block)

%endfunction
```

Listing A.1: Matlab code for the S-Function Input Block

A.2 S-Function ModelOutput

```
function ModelOutput(block)

%MSFUNTMPL A template for an M-file S-function

%   The M-file S-function is written as a MATLAB function with the

%   same name as the S-function.

%

%   It should be noted that the M-file S-function is very similar

%   to Level-2 C-Mex S-functions. You should be able to get more

%   information for each of the block methods by referring to the

%   documentation for C-Mex S-functions.

%

%   Copyright 2003-2006 The MathWorks, Inc.

%   $Revision: 1.1.6.13 $



%%

%% The setup method is used to setup the basic attributes of the

%% S-function such as ports, parameters, etc. Do not add any other

%% calls to the main body of the function.

%%

setup(block);



%endfunction



%% Function: setup ===================================================

%% Abstract:

%%   Set up the S-function block's basic characteristics such as:

%%   - Input ports

%%   - Output ports

%%   - Dialog parameters

%%   - Options

%%

%%   Required         : Yes

%%   C-Mex counterpart: mdlInitializeSizes

%%

function setup(block)



  % Register number of ports

  nu = 1; % 0 input ports

  ny = 0; % 1 output port



  block.NumInputPorts  = nu;

  block.NumOutputPorts = ny;



  block.NumContStates = 0;



  % Setup port properties to be inherited or dynamic

  block.SetPreCompOutPortInfoToDynamic;



  for i = 1:nu
```

```

 %block.InputPort(1).Dimensions = size(1);

% block.OutputPort(1).SamplingMode = 'sample';

 block.InputPort(1).DatatypeID  = 0; % double

 block.InputPort(1).Complexity  = 'Real';

 %block.OutputPort(1).DimensionsMode = 'Variable';

end



% Register parameters

block.NumDialogPrms     = 1;



block.DialogPrmsTunable = {'Nontunable'};



% Register sample times

%  [0 offset]            : Continuous sample time

%  [positive_num offset] : Discrete sample time

%

%  [-1, 0]               : Inherited sample time

%  [-2, 0]               : Variable sample time

block.SampleTimes = [-1 0];



%% -----------------------------------------------------------------

%% Options

%% -----------------------------------------------------------------

% Specify if Accelerator should use TLC or call back into

```

```
  % M-file

 % block.SetAccelRunOnTLC(false);




  %% ----------------------------------------------------------------

  %% Register callback methods

  %% ----------------------------------------------------------------

  %block.RegBlockMethod('PostPropagationSetup',    @DoPostPropSetup);

  %block.RegBlockMethod('CheckParameters', @CheckPrms); % allow validation of block
      's dialog parameters

  %block.RegBlockMethod('ProcessParameters', @ProcessPrms); % Called in order to
      allow update of run-time parameters

  %block.RegBlockMethod('InitializeConditions', @InitializeConditions); % Called in
       order to initialize state and work area values

  %block.RegBlockMethod('Start', @Start);

  block.RegBlockMethod('Outputs', @Outputs); % Called to generate block outputs in
      simulation step





%% Matlab callbacks:



function DoPostPropSetup(block)



  %% Setup Dwork

  block.NumDworks = 1;



  %% [Slice maxSlice]

  block.Dwork(1).Name         = 'inputIndex';

  block.Dwork(1).Dimensions   = 1;

  block.Dwork(1).DatatypeID   = 7;

  block.Dwork(1).Complexity   = 'Real';



%endfunction



function CheckPrms(block)

%endfunction



function ProcessPrms(block)



%  block.AutoUpdateRuntimePrms;

%  block.OutputPort(opIdx).CurrentDimensions = 1;





%endfunction

function InitializeConditions(block)

%  block.Dwork(1).Data = 1;



%endfunction



function Start(block)



```

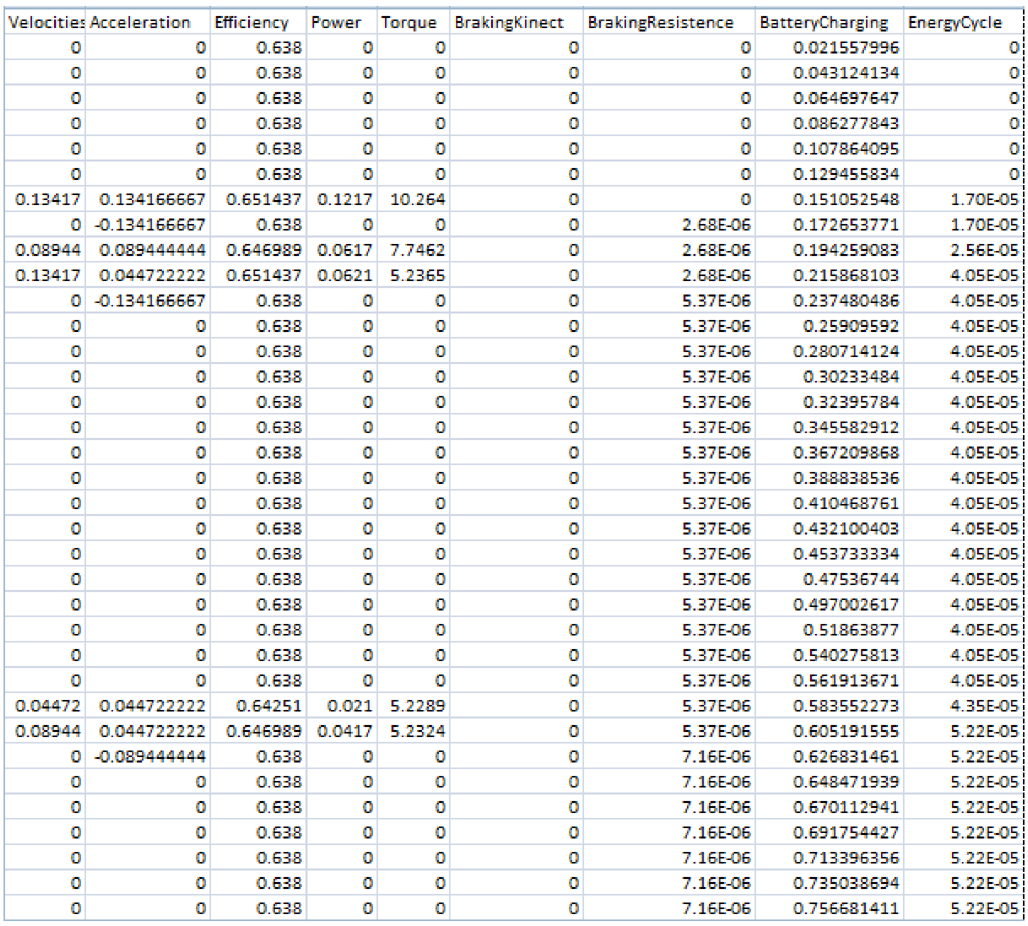
```
  %% Initialize Dwork

%  block.Dwork(1).Data = 1;

%endfunction



function Outputs(block)

%   block.OutputPort(1).Data = evalin('base', block.DialogPrm(1).Data)

%    block.OutputPort(1).Data = evalin('base', 'uWorkspace');

   assignin('base', block.DialogPrm(1).Data, block.InputPort(1).Data(1));



%endfunction




function Terminate(block)

%endfunction
```

Listing A.2: Matlab code for the S-Function Output Block

Appendix B

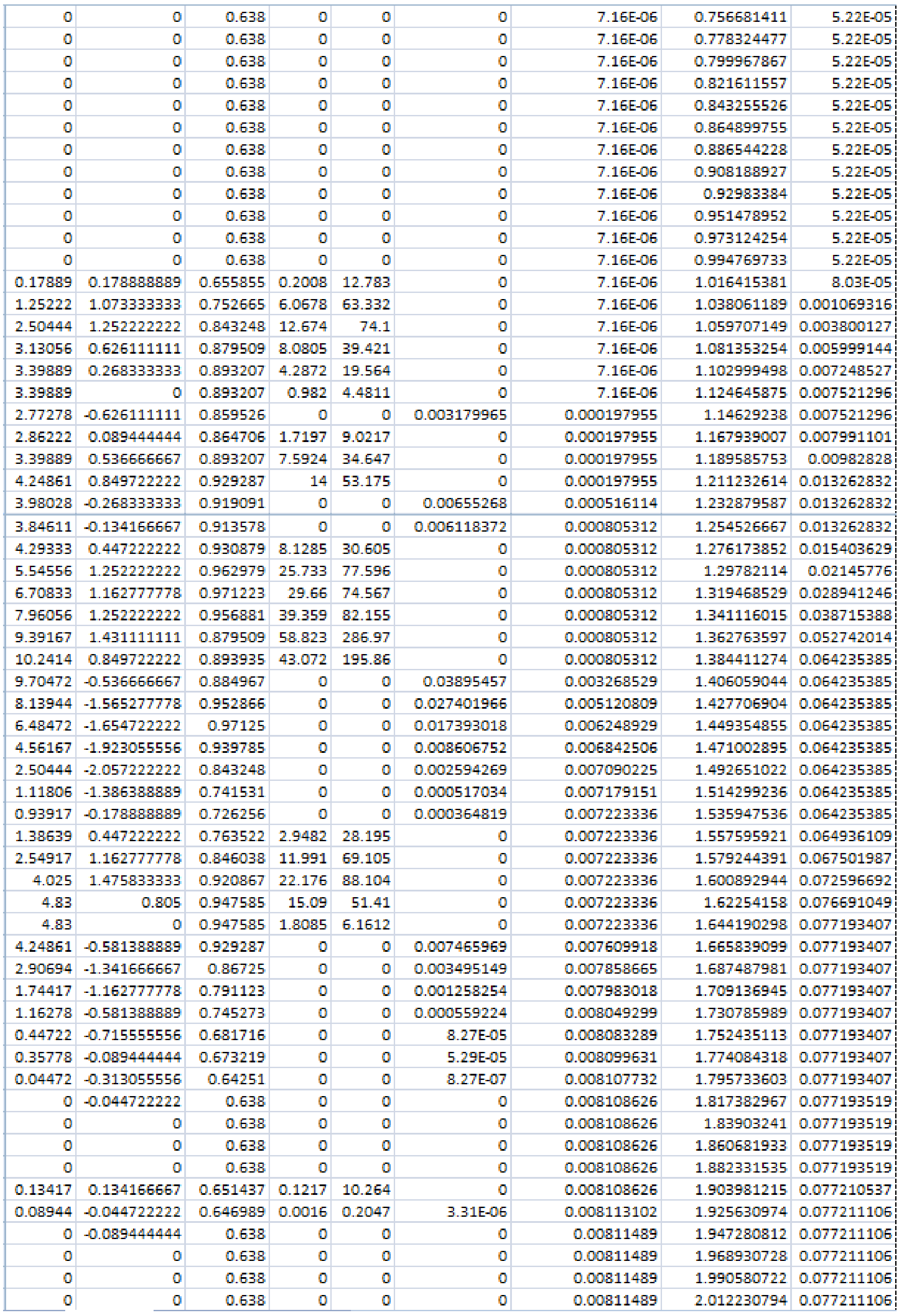
Field Experiments Results

The information presented next are the results of experiments performed on the field. In Figures B.1, B.2 and B.3 are illustrated the results of the different EBPS metrics for a given set of velocities.



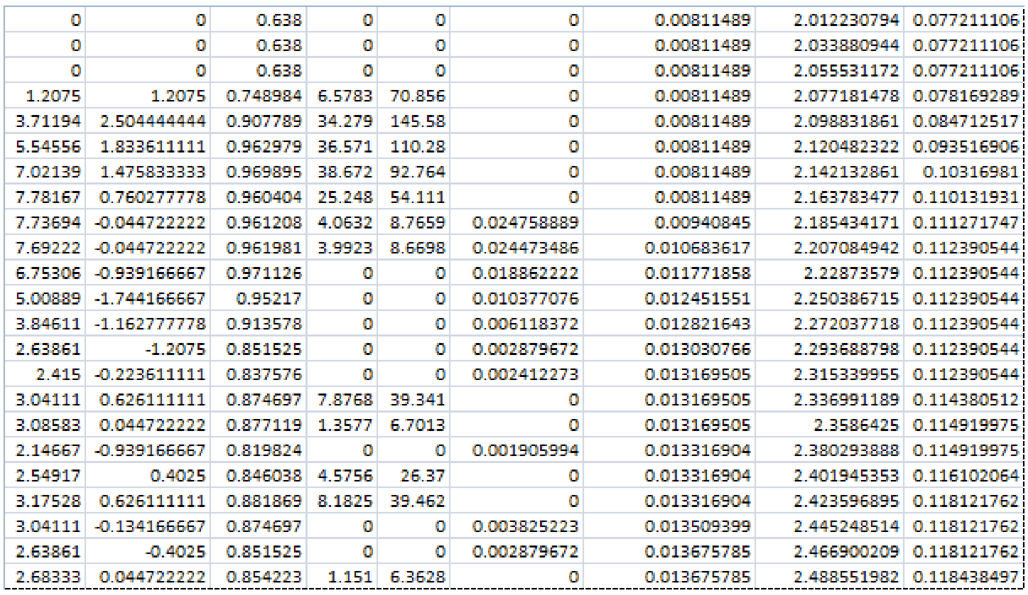
Velocities	Acceleration	Efficiency	Power	Torque	BrakingKinect	BrakingResistence	BatteryCharging	EnergyCycle
0	0	0.638	0	0	0	0	0.021557996	0
0	0	0.638	0	0	0	0	0.043124134	0
0	0	0.638	0	0	0	0	0.064697647	0
0	0	0.638	0	0	0	0	0.086277843	0
0	0	0.638	0	0	0	0	0.107864095	0
0	0	0.638	0	0	0	0	0.129455834	0
0.13417	0.134166667	0.651437	0.1217	10.264	0	0	0.151052548	1.70E-05
0	-0.134166667	0.638	0	0	0	2.68E-06	0.172653771	1.70E-05
0.08944	0.089444444	0.646989	0.0617	7.7462	0	2.68E-06	0.194259083	2.56E-05
0.13417	0.044722222	0.651437	0.0621	5.2365	0	2.68E-06	0.215868103	4.05E-05
0	-0.134166667	0.638	0	0	0	5.37E-06	0.237480486	4.05E-05
0	0	0.638	0	0	0	5.37E-06	0.25909592	4.05E-05
0	0	0.638	0	0	0	5.37E-06	0.280714124	4.05E-05
0	0	0.638	0	0	0	5.37E-06	0.30233484	4.05E-05
0	0	0.638	0	0	0	5.37E-06	0.32395784	4.05E-05
0	0	0.638	0	0	0	5.37E-06	0.345582912	4.05E-05
0	0	0.638	0	0	0	5.37E-06	0.367209868	4.05E-05
0	0	0.638	0	0	0	5.37E-06	0.388838536	4.05E-05
0	0	0.638	0	0	0	5.37E-06	0.410468761	4.05E-05
0	0	0.638	0	0	0	5.37E-06	0.432100403	4.05E-05
0	0	0.638	0	0	0	5.37E-06	0.453733334	4.05E-05
0	0	0.638	0	0	0	5.37E-06	0.47536744	4.05E-05
0	0	0.638	0	0	0	5.37E-06	0.497002617	4.05E-05
0	0	0.638	0	0	0	5.37E-06	0.51863877	4.05E-05
0	0	0.638	0	0	0	5.37E-06	0.540275813	4.05E-05
0	0	0.638	0	0	0	5.37E-06	0.561913671	4.05E-05
0.04472	0.044722222	0.64251	0.021	5.2289	0	5.37E-06	0.583552273	4.35E-05
0.08944	0.044722222	0.646989	0.0417	5.2324	0	5.37E-06	0.605191555	5.22E-05
0	-0.089444444	0.638	0	0	0	7.16E-06	0.626831461	5.22E-05
0	0	0.638	0	0	0	7.16E-06	0.648471939	5.22E-05
0	0	0.638	0	0	0	7.16E-06	0.670112941	5.22E-05
0	0	0.638	0	0	0	7.16E-06	0.691754427	5.22E-05
0	0	0.638	0	0	0	7.16E-06	0.713396356	5.22E-05
0	0	0.638	0	0	0	7.16E-06	0.735038694	5.22E-05
0	0	0.638	0	0	0	7.16E-06	0.756681411	5.22E-05

Figure B.1: Field Experiment Results from EBPS



0	0	0.638	0	0	0	7.16E-06	0.756681411	5.22E-05
0	0	0.638	0	0	0	7.16E-06	0.778324477	5.22E-05
0	0	0.638	0	0	0	7.16E-06	0.799967867	5.22E-05
0	0	0.638	0	0	0	7.16E-06	0.821611557	5.22E-05
0	0	0.638	0	0	0	7.16E-06	0.843255526	5.22E-05
0	0	0.638	0	0	0	7.16E-06	0.864899755	5.22E-05
0	0	0.638	0	0	0	7.16E-06	0.886544228	5.22E-05
0	0	0.638	0	0	0	7.16E-06	0.908188927	5.22E-05
0	0	0.638	0	0	0	7.16E-06	0.92983384	5.22E-05
0	0	0.638	0	0	0	7.16E-06	0.951478952	5.22E-05
0	0	0.638	0	0	0	7.16E-06	0.973124254	5.22E-05
0	0	0.638	0	0	0	7.16E-06	0.994769733	5.22E-05
0.17889	0.178888889	0.655855	0.2008	12.783	0	7.16E-06	1.016415381	8.03E-05
1.25222	1.073333333	0.752665	6.0678	63.332	0	7.16E-06	1.038061189	0.001069316
2.50444	1.252222222	0.843248	12.674	74.1	0	7.16E-06	1.059707149	0.003800127
3.13056	0.626111111	0.879509	8.0805	39.421	0	7.16E-06	1.081353254	0.005999144
3.39889	0.268333333	0.893207	4.2872	19.564	0	7.16E-06	1.102999498	0.007248527
3.39889	0	0.893207	0.982	4.4811	0	7.16E-06	1.124645875	0.007521296
2.77278	-0.626111111	0.859526	0	0	0.003179965	0.000197955	1.14629238	0.007521296
2.86222	0.089444444	0.864706	1.7197	9.0217	0	0.000197955	1.167939007	0.007991101
3.39889	0.536666667	0.893207	7.5924	34.647	0	0.000197955	1.189585753	0.00982828
4.24861	0.849722222	0.929287	14	53.175	0	0.000197955	1.211232614	0.013262832
3.98028	-0.268333333	0.919091	0	0	0.00655268	0.000516114	1.232879587	0.013262832
3.84611	-0.134166667	0.913578	0	0	0.006118372	0.000805312	1.254526667	0.013262832
4.29333	0.447222222	0.930879	8.1285	30.605	0	0.000805312	1.276173852	0.015403629
5.54556	1.252222222	0.962979	25.733	77.596	0	0.000805312	1.29782114	0.02145776
6.70833	1.162777778	0.971223	29.66	74.567	0	0.000805312	1.319468529	0.028941246
7.96056	1.252222222	0.956881	39.359	82.155	0	0.000805312	1.341116015	0.038715388
9.39167	1.431111111	0.879509	58.823	286.97	0	0.000805312	1.362763597	0.052742014
10.2414	0.849722222	0.893935	43.072	195.86	0	0.000805312	1.384411274	0.064235385
9.70472	-0.536666667	0.884967	0	0	0.03895457	0.003268529	1.406059044	0.064235385
8.13944	-1.565277778	0.952866	0	0	0.027401966	0.005120809	1.427706904	0.064235385
6.48472	-1.654722222	0.97125	0	0	0.017393018	0.006248929	1.449354855	0.064235385
4.56167	-1.923055556	0.939785	0	0	0.008606752	0.006842506	1.471002895	0.064235385
2.50444	-2.057222222	0.843248	0	0	0.002594269	0.007090225	1.492651022	0.064235385
1.11806	-1.386388889	0.741531	0	0	0.000517034	0.007179151	1.514299236	0.064235385
0.93917	-0.178888889	0.726256	0	0	0.000364819	0.007223336	1.535947536	0.064235385
1.38639	0.447222222	0.763522	2.9482	28.195	0	0.007223336	1.557595921	0.064936109
2.54917	1.162777778	0.846038	11.991	69.105	0	0.007223336	1.579244391	0.067501987
4.025	1.475833333	0.920867	22.176	88.104	0	0.007223336	1.600892944	0.072596692
4.83	0.805	0.947585	15.09	51.41	0	0.007223336	1.62254158	0.076691049
4.83	0	0.947585	1.8085	6.1612	0	0.007223336	1.644190298	0.077193407
4.24861	-0.581388889	0.929287	0	0	0.007465969	0.007609918	1.665839099	0.077193407
2.90694	-1.341666667	0.86725	0	0	0.003495149	0.007858665	1.687487981	0.077193407
1.74417	-1.162777778	0.791123	0	0	0.001258254	0.007983018	1.709136945	0.077193407
1.16278	-0.581388889	0.745273	0	0	0.000559224	0.008049299	1.730785989	0.077193407
0.44722	-0.715555556	0.681716	0	0	8.27E-05	0.008083289	1.752435113	0.077193407
0.35778	-0.089444444	0.673219	0	0	5.29E-05	0.008099631	1.774084318	0.077193407
0.04472	-0.313055556	0.64251	0	0	8.27E-07	0.008107732	1.795733603	0.077193407
0	-0.044722222	0.638	0	0	0	0.008108626	1.817382967	0.077193519
0	0	0.638	0	0	0	0.008108626	1.83903241	0.077193519
0	0	0.638	0	0	0	0.008108626	1.860681933	0.077193519
0	0	0.638	0	0	0	0.008108626	1.882331535	0.077193519
0.13417	0.134166667	0.651437	0.1217	10.264	0	0.008108626	1.903981215	0.077210537
0.08944	-0.044722222	0.646989	0.0016	0.2047	3.31E-06	0.008113102	1.925630974	0.077211106
0	-0.089444444	0.638	0	0	0	0.00811489	1.947280812	0.077211106
0	0	0.638	0	0	0	0.00811489	1.968930728	0.077211106
0	0	0.638	0	0	0	0.00811489	1.990580722	0.077211106
0	0	0.638	0	0	0	0.00811489	2.012230794	0.077211106

Figure B.2: Field Experiment Results from EBPS (continuation)



0	0	0.638	0	0	0	0.00811489	2.012230794	0.077211106
0	0	0.638	0	0	0	0.00811489	2.033880944	0.077211106
0	0	0.638	0	0	0	0.00811489	2.055531172	0.077211106
1.2075	1.2075	0.748984	6.5783	70.856	0	0.00811489	2.077181478	0.078169289
3.71194	2.504444444	0.907789	34.279	145.58	0	0.00811489	2.098831861	0.084712517
5.54556	1.833611111	0.962979	36.571	110.28	0	0.00811489	2.120482322	0.093516906
7.02139	1.475833333	0.969895	38.672	92.764	0	0.00811489	2.142132861	0.10316981
7.78167	0.760277778	0.960404	25.248	54.111	0	0.00811489	2.163783477	0.110131931
7.73694	-0.044722222	0.961208	4.0632	8.7659	0.024758889	0.00940845	2.185434171	0.111271747
7.69222	-0.044722222	0.961981	3.9923	8.6698	0.024473486	0.010683617	2.207084942	0.112390544
6.75306	-0.939166667	0.971126	0	0	0.018862222	0.011771858	2.22873579	0.112390544
5.00889	-1.744166667	0.95217	0	0	0.010377076	0.012451551	2.250386715	0.112390544
3.84611	-1.162777778	0.913578	0	0	0.006118372	0.012821643	2.272037718	0.112390544
2.63861	-1.2075	0.851525	0	0	0.002879672	0.013030766	2.293688798	0.112390544
2.415	-0.223611111	0.837576	0	0	0.002412273	0.013169505	2.315339955	0.112390544
3.04111	0.626111111	0.874697	7.8768	39.341	0	0.013169505	2.336991189	0.114380512
3.08583	0.044722222	0.877119	1.3577	6.7013	0	0.013169505	2.3586425	0.114919975
2.14667	-0.939166667	0.819824	0	0	0.001905994	0.013316904	2.380293888	0.114919975
2.54917	0.4025	0.846038	4.5756	26.37	0	0.013316904	2.401945353	0.116102064
3.17528	0.626111111	0.881869	8.1825	39.462	0	0.013316904	2.423596895	0.118121762
3.04111	-0.134166667	0.874697	0	0	0.003825223	0.013509399	2.445248514	0.118121762
2.63861	-0.4025	0.851525	0	0	0.002879672	0.013675785	2.466900209	0.118121762
2.68333	0.044722222	0.854223	1.151	6.3628	0	0.013675785	2.488551982	0.118438497

Figure B.3: Field Experiment Results from EBPS (continuation)

Field Experiments Results

Appendix C

Federation Object Model (FOM) Specification for Electric Bus in Traffic Simulation Federation

The information oi this appendix focus on giving the Federation Object Model specification for the project federation and so, the necessary knowledge to reproduce or modify it.

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<objectModel xsi:schemaLocation="http://standards.ieee.org/IEEE1516-2010 http://
    standards.ieee.org/downloads/1516/1516.2-2010/IEEE1516-DIF-2010.xsd" xmlns="
    http://standards.ieee.org/IEEE1516-2010" xmlns:xsi="http://www.w3.org/2001/
    XMLSchema-instance">
    <modelIdentification>
        <name>Perfromance Statistics FOM</name>
        <type>FOM</type>
        <version>1.0</version>
        <modificationDate>2012-12-04</modificationDate>
        <securityClassification>Unclassified</securityClassification>
        <purpose>Master Thesis</purpose>
        <applicationDomain>Engineering</applicationDomain>
        <description>FOM for Integration between SUMO and Simulink.</description>
        <useLimitation>None</useLimitation>
        <poc>
            <pocType>Primary author</pocType>
            <pocName>Jose Macedo</pocName>
            <pocOrg>FEUP</pocOrg>
            <pocTelephone>+351916963555</pocTelephone>
            <pocEmail>jose.macedo@fe.up.pt</pocEmail>
        </poc>
        <reference>
            <type>Document</type>
            <identification>EBPS with SUMO (Sep 2012)</identification>
        </reference>
        <other></other>
    </modelIdentification>
    <objects>
        <objectClass>
            <name>HLAobjectRoot</name>
            <objectClass>
                <name>Bus</name>
                <sharing>PublishSubscribe</sharing>
                <semantics>A bus for the Performance Statistics federation</
                    semantics>
                <attribute>
                    <name>Name</name>
                    <dataType>HLAunicodeString</dataType>
                    <updateType>Static</updateType>
                    <updateCondition>NA</updateCondition>
                    <ownership>NoTransfer</ownership>
                    <sharing>PublishSubscribe</sharing>
                    <dimensions/>
                    <transportation>HLAreliable</transportation>
                    <order>Receive</order>
                    <semantics>Name of the bus</semantics>
                </attribute>
                <attribute>
                    <name>Velocity</name>
                    <dataType>VelocityFloat64</dataType>
                    <updateType>Conditional</updateType>
                    <updateCondition>On change</updateCondition>
                    <ownership>NoTransfer</ownership>
                    <sharing>PublishSubscribe</sharing>
                    <dimensions/>
                    <transportation>HLAreliable</transportation>
                    <order>Receive</order>
                    <semantics>Current velocity of the bus at each point in time</
                        semantics>
                </attribute>
                <attribute>
                    <name>Acceleration</name>
                    <dataType>AccelerationFloat64</dataType>
                    <updateType>Conditional</updateType>
                    <updateCondition>On change</updateCondition>
                    <ownership>NoTransfer</ownership>
                    <sharing>PublishSubscribe</sharing>
                    <dimensions/>
                    <transportation>HLAreliable</transportation>
                    <order>Receive</order>
                    <semantics>Current acceleration of the bus at each point in
                        time</semantics>
                </attribute>
                <attribute>
                    <name>Power</name>
                    <dataType>PowerFloat64</dataType>
                    <updateType>Conditional</updateType>
                    <updateCondition>On change</updateCondition>
                    <ownership>NoTransfer</ownership>
                    <sharing>PublishSubscribe</sharing>
                    <dimensions/>
                    <transportation>HLAreliable</transportation>
                    <order>Receive</order>
                    <semantics>Current power of the bus at each point in time</
                        semantics>
                </attribute>
                <attribute>
                    <name>Torque</name>
                    <dataType>TorqueFloat64</dataType>
                    <updateType>Conditional</updateType>
                    <updateCondition>On change</updateCondition>
                    <ownership>NoTransfer</ownership>
                    <sharing>PublishSubscribe</sharing>
                    <dimensions/>
                    <transportation>HLAreliable</transportation>
                    <order>Receive</order>
                    <semantics>Current torque of the bus at each point in time</
                        semantics>
                </attribute>
                <attribute>
                    <name>Efficiency</name>
                    <dataType>EfficiencyFloat64</dataType>
                    <updateType>Conditional</updateType>
                    <updateCondition>On change</updateCondition>
                    <ownership>NoTransfer</ownership>
                    <sharing>PublishSubscribe</sharing>
                    <dimensions/>
                    <transportation>HLAreliable</transportation>
                    <order>Receive</order>
                    <semantics>Current efficiency of the bus at each point in time<
                        /semantics>
                </attribute>
                <attribute>
                    <name>TotalCycleEnergy</name>
                    <dataType>TotalCycleEnergyFloat64</dataType>
                    <updateType>Conditional</updateType>
                    <updateCondition>On change</updateCondition>
                    <ownership>NoTransfer</ownership>
                    <sharing>PublishSubscribe</sharing>
                    <dimensions/>
                    <transportation>HLAreliable</transportation>
                    <order>Receive</order>
                    <semantics>Current total cycle energy of the bus at each point in time</semantics>
                </attribute>
                <attribute>
                    <name>BrakingKinectEnergy</name>
                    <dataType>BrakingKinectEnergyFloat64</dataType>
                    <updateType>Conditional</updateType>
                    <updateCondition>On change</updateCondition>
                    <ownership>NoTransfer</ownership>
                    <sharing>PublishSubscribe</sharing>
                    <dimensions/>
                    <transportation>HLAreliable</transportation>
                    <order>Receive</order>
                    <semantics>Current braking kinect energy of the bus at each point in time</semantics>
                </attribute>
                <attribute>
                    <name>BrakingResistanceEnergy</name>
                    <dataType>BrakingResistanceEnergyFloat64</dataType>
                    <updateType>Conditional</updateType>
                    <updateCondition>On change</updateCondition>
                    <ownership>NoTransfer</ownership>
                    <sharing>PublishSubscribe</sharing>
                    <dimensions/>
                    <transportation>HLAreliable</transportation>
                    <order>Receive</order>
                    <semantics>Current braking resistance energy of the bus at each point in time</semantics>
                </attribute>
                <attribute>
                    <name>SuperCapacitorsChargingEnergy</name>
                    <dataType>SupercapacitorsChargingEnergyFloat64</dataType>
                    <updateType>Conditional</updateType>
                    <updateCondition>On change</updateCondition>
                    <ownership>NoTransfer</ownership>
                    <sharing>PublishSubscribe</sharing>
                    <dimensions/>
                    <transportation>HLAreliable</transportation>
                    <order>Receive</order>
                    <semantics>Current supercapacitors charging energy of the bus at each point in time</semantics>
                </attribute>
                <attribute>
                    <name>SuperCapacitorsDischargingEnergy</name>
                    <dataType>SupercapacitorsDischargingEnergyFloat64</dataType>
                    <updateType>Conditional</updateType>
                    <updateCondition>On change</updateCondition>
                    <ownership>NoTransfer</ownership>
                    <sharing>PublishSubscribe</sharing>
```

```xml
                    <dimensions/>
                    <transportation>HLAreliable</transportation>
                    <order>Receive</order>
                    <semantics>Current supercapacitors discharging energy of the bus at each point in time</semantics>
                </attribute>
                <attribute>
                    <name>BatteriesChargingEnergy</name>
                    <dataType>BatteriesChargingEnergyFloat64</dataType>
                    <updateType>Conditional</updateType>
                    <updateCondition>On change</updateCondition>
                    <ownership>NoTransfer</ownership>
                    <sharing>PublishSubscribe</sharing>
                    <dimensions/>
                    <transportation>HLAreliable</transportation>
                    <order>Receive</order>
                    <semantics>Current batteries charging energy of the bus at each point in time</semantics>
                </attribute>
            </objectClass>
        </objectClass>
    </objects>
    <interactions>
        <interactionClass>
            <name>HLAinteractionRoot</name>
            <interactionClass>
                <name>Start</name>
                <sharing>PublishSubscribe</sharing>
                <dimensions/>
                <transportation>HLAreliable</transportation>
                <order>Receive</order>
                <semantics>Interaction to Start the simulation</semantics>
                <parameter>
                    <name>TimeScaleFactor</name>
                    <dataType>ScaleFactorFloat32</dataType>
                    <semantics>How fast will the simulation run compared to real time. Example: 1.0= real time, 2.0 indicates that the simulation runs at twice the speed. </semantics>
                </parameter>
            </interactionClass>
            <interactionClass>
                <name>Stop</name>
                <sharing>PublishSubscribe</sharing>
                <dimensions/>
                <transportation>HLAreliable</transportation>
                <order>Receive</order>
                <semantics>Interaction to Stop the simulation</semantics>
            </interactionClass>
        </interactionClass>
    </interactions>
    <switches>
        <autoProvide isEnabled="true"/>
        <conveyRegionDesignatorSets isEnabled="false"/>
        <conveyProducingFederate isEnabled="false"/>
        <attributeScopeAdvisory isEnabled="false"/>
        <attributeRelevanceAdvisory isEnabled="false"/>
        <objectClassRelevanceAdvisory isEnabled="false"/>
        <interactionRelevanceAdvisory isEnabled="false"/>
        <serviceReporting isEnabled="false"/>
        <exceptionReporting isEnabled="false"/>
        <delaySubscriptionEvaluation isEnabled="false"/>
        <automaticResignAction resignAction="CancelThenDeleteThenDivest"/>
    </switches>
    <dataTypes>
        <simpleDataTypes>
            <simpleData>
                <name>VelocityFloat64</name>
                <representation>HLAfloat64BE</representation>
                <units>Meters per second</units>
                <resolution>0.000001</resolution>
                <accuracy>0.000001</accuracy>
                <semantics>Double that describes the velocity.</semantics>
            </simpleData>
            <simpleData>
                <name>ScaleFactorFloat32</name>
                <representation>HLAfloat32BE</representation>
                <units>NA</units>
                <resolution>0.001</resolution>
                <accuracy>0.001</accuracy>
                <semantics>Ratio between two values. Used for the scaling of time
                    or space. Negative numbers are not allowed.
</semantics>
            </simpleData>
            <simpleData>
                <name>AccelerationFloat64</name>
                <representation>HLAfloat64BE</representation>
                <units>ND</units>
                <resolution>0.000001</resolution>
                <accuracy>0.000001</accuracy>
                <semantics>Double that describes the acceleration.</semantics>
            </simpleData>
            <simpleData>
                <name>TorqueFloat64</name>
                <representation>HLAfloat64BE</representation>
                <units>ND</units>
                <resolution>0.000001</resolution>
                <accuracy>0.000001</accuracy>
                <semantics>Double that describes the torque.</semantics>
            </simpleData>
            <simpleData>
                <name>EfficiencyFloat64</name>
                <representation>HLAfloat64BE</representation>
                <units>ND</units>
                <resolution>0.000001</resolution>
                <accuracy>0.000001</accuracy>
                <semantics>Double that describes the efficiency.</semantics>
            </simpleData>
            <simpleData>
                <name>PowerFloat64</name>
                <representation>HLAfloat64BE</representation>
                <units>ND</units>
                <resolution>0.000001</resolution>
                <accuracy>0.000001</accuracy>
                <semantics>Double that describes the power.</semantics>
            </simpleData>
            <simpleData>
                <name>TotalCycleEnergyFloat64</name>
                <representation>HLAfloat64BE</representation>
                <units>ND</units>
                <resolution>0.000001</resolution>
                <accuracy>0.000001</accuracy>
                <semantics>Double that describes the TotalCycleEnergy.</semantics>
            </simpleData>
            <simpleData>
                <name>BrakingKinectEnergyFloat64</name>
                <representation>HLAfloat64BE</representation>
                <units>ND</units>
                <resolution>0.000001</resolution>
                <accuracy>0.000001</accuracy>
                <semantics>Double that describes the BrakingKinectEnergy.</semantics>
            </simpleData>
            <simpleData>
                <name>BrakingResistanceEnergyFloat64</name>
                <representation>HLAfloat64BE</representation>
                <units>ND</units>
                <resolution>0.000001</resolution>
                <accuracy>0.000001</accuracy>
                <semantics>Double that describes the BrakingResistanceEnergy.</semantics>
            </simpleData>
            <simpleData>
                <name>SupercapacitorsChargingEnergyFloat64</name>
                <representation>HLAfloat64BE</representation>
                <units>ND</units>
                <resolution>0.000001</resolution>
                <accuracy>0.000001</accuracy>
                <semantics>Double that describes the SupercapacitorsChargingEnergy.</semantics>
            </simpleData>
            <simpleData>
                <name>SupercapacitorsDischargingEnergyFloat64</name>
                <representation>HLAfloat64BE</representation>
                <units>ND</units>
                <resolution>0.000001</resolution>
                <accuracy>0.000001</accuracy>
                <semantics>Double that describes the SupercapacitorsDischargingEnergy.</semantics>
            </simpleData>
            <simpleData>
                <name>BatteriesChargingEnergyFloat64</name>
                <representation>HLAfloat64BE</representation>
                <units>ND</units>
                <resolution>0.000001</resolution>
                <accuracy>0.000001</accuracy>
                <semantics>Double that describes the BatteriesChargingEnergy.</semantics>
            </simpleData>
        </simpleDataTypes>
    </dataTypes>
    <notes>
        <note>
            <label>parameters</label>
            <semantics>Consider using a float for this for higher accuracy</semantics>
        </note>
    </notes>
</objectModel>
```

Listing C.1: FOM specification