

Eye-tracking for designing the user interface of a multimodal train driving simulator and for evaluating its usage by learners

A structured approach and a software environment

Dissertation presented by
Jonathan POWELL

for obtaining the Master's degree in
Computer Science Engineering

Supervisor(s)
Jean VANDERDONCKT

Reader(s)
Laurent DE PLAEN, Jorge Luis PEREZ MEDINA , Kieffer SUZANNE

Academic year 2016-2017

Acknowledgements

As a preface to this dissertation, I would like to express my sincere thanks to the people who helped me and contributed to the development of this dissertation and allowed for the success of this last academic year.

First of all, I would particularly like to thank my dissertation supervisor Professor Jean VANDERDONCKT for his valuable advice and constructive comments, as well as for the time and the listening he has given me over the past year.

I would like to sincerely thank the technical manager of Transurb Simulation Laurent DE PLAEN who was my internship supervisor during my entire training within the company. Who guided me during my internship and took the time to find solutions to progress in my work. I also thank him for the confidence that he has placed in me since the beginning of the project.

A big thank you is addressed to the whole Transurb Simulation team, for their warm-hearted welcome, for the constructive exchanges we have had and for the time they have devoted to me during the internship related to this master dissertation.

Finally, I want to express a special thanks to my family and friends who have always supported and encouraged me during the realisation of this dissertation, but also for the patience they have shown throughout this assessment.

Also, I wish to sincerely thank all those who have contributed directly or indirectly in supporting me for this project.

I would like to dedicate this dissertation to the memory of Françoise GODFRIAUX (1962-2017) and to my loving grandmother Anne LE RICHE (1931-2017) whom both recently passed away at the time of writing these lines. My thoughts go especially to the family of Françoise GODFRIAUX who taught me to never stop fighting and who will always inspire me with this thirst for knowledge.

Contents

1	Introduction	1
1.1	Context of the Problem	1
1.2	Central Statement of the Dissertation	2
1.3	Coverage Assumptions and Concepts	2
1.4	Content of the Dissertation	2
2	Train Simulation Overview	5
2.1	Computer Simulation	5
2.1.1	Formal Definition	5
2.1.2	Types of Computer Simulations	6
2.1.3	Purposes of the Simulation	6
2.2	Understanding the Railway Simulation Motivations	7
2.2.1	Railway Market Survey the Social and Economic Issues	7
2.3	Transurb Simulation for Train Simulator	9
2.3.1	Solutions of Transurb Simulation	9
2.3.2	Existing software Modules	11
2.3.3	Driving Desktop Simulator System	12
2.3.4	DB SyncManager Module	13
2.3.5	SyncManager user interface	15
2.3.6	DB SyncManager Intent and Enhancement	16
2.3.7	Advanced Training Solutions	17
2.4	Training Simulation Future Development	18
3	Eye-tracking an Emerging Technology	21
3.1	Eye Tracking Terminology	21
3.2	State of the Art of Current Applications	23
3.2.1	Eye Tracking for User Evaluation	23
3.2.2	Eye Tracking for User Interaction	26
3.3	State of the Art Analyses	27
3.4	EyeWorks and the Fovio Eye Tracker	28
3.5	Usability Evaluation of Real Rail Road	29
3.5.1	Procedure to Conduct an Eye Tracking HCI Evaluation Experiment	31
4	Usybus Software Environment for HCI Study	33
4.1	Usybus Multi Agents Architecture	33
4.2	Existing Usybus Agents	35
4.2.1	The Usybus Eye Tracking Monitor	35
4.2.2	The Usybus Pupils Sizes Monitor	36
4.2.3	The Usybus Eye Tracking Analyzer	36
4.2.4	The Usybus Controller Agents	37
4.2.5	The Usybus Messages Recorder / Player	38

4.3	Usybus EyeWorks Controller agent	38
4.4	Usybus Cognitive Workload Monitor Agent	40
4.5	Upgrade of the Usybus Eye Tracking Monitor Agent	41
4.6	Usybus data types	42
4.6.1	The Cognitive Workload Type	45
4.6.2	The Task type	45
4.6.3	Types usage overview	45
4.7	AUML Formalisation for Agents Specification	46
4.8	User Interface Adaptation as Usybus Agent	47
4.8.1	The PDA-LPA Design Space	48
5	UsiTracker for Conception and Evaluation of User Interface	51
5.1	Designing the User Interface	51
5.1.1	XML Representation for User Interface Templates	53
5.2	Evaluating the User Interface	57
5.2.1	Usybus Widget Naming Standard	57
5.3	Recording the UsiTracker Session	59
5.3.1	The Visual Attention for Multi-screen	60
5.4	Usitracker for User Interface Design Evaluation Cycle	61
5.5	UsiTracker Test Case with SmallApp Template	62
6	Conclusion and Perspectives	67
6.1	Summary of the Problematic	67
6.2	Pros and Cons of the UsiTracker Tool	68
6.3	Improvements and Perspective	68
6.3.1	The Usybus Architecture	68
6.3.2	The UsiTracker Softwares	68
6.3.3	The Graph Analyser	69
	References	69
	Appendices	77
A	Hardware specification	79
A.1	The Eye Tribe device	79
A.2	The Tobii 50 Series devices	79
A.2.1	The Tobii 1750	80
A.2.2	The Tobii X50	80
A.2.3	The Tobii T60 and T120	81
A.2.4	The Tobii X60 and X120	82
B	The Usybus Widget Tracking Library	83
B.1	Supported Widget Type	83

Acronyms

AR Augmented Reality. [18](#)

AUI Adaptive User Interface. [47](#), [48](#)

AUML Agent Unified Modeling Language. [46](#)

CSV Comma-Separated Values. [59](#)

DDS Driving Desktop Simulator. [12](#), [13](#), [14](#), [15](#), [16](#), [17](#)

FIPA Foundation for Intelligent Physical Agents. [46](#)

HCI Human-Computer Interaction. [2](#), [3](#), [18](#), [21](#), [27](#), [33](#), [34](#), [47](#), [51](#), [61](#), [67](#), [68](#)

HITL Human-in-the-Loop. [6](#)

I-DT Dispersion-Threshold Identification. [36](#)

ICA Index of Cognitive Activity. [22](#), [23](#), [29](#), [39](#), [40](#), [45](#), [60](#), [62](#), [64](#), [68](#), [69](#)

IT Information Technology. [1](#), [67](#)

JAR Java Archive. [35](#)

JSON JavaScript Object Notation. [34](#)

JVM Java Virtual Machine. [35](#)

LAN Local Area Network. [17](#)

LIG Laboratoire d'Informatique de Grenoble. [36](#)

LPA Learning-Prediction-Action. [48](#)

OS Operating System. [35](#)

PDA Perception-Decision-Action. [48](#)

R³ Real Rail Road. [12](#), [13](#), [14](#), [16](#), [29](#), [31](#), [67](#), [69](#)

UI User Interface. [2](#), [3](#), [28](#), [29](#), [33](#), [47](#), [48](#), [51](#), [52](#), [53](#), [55](#), [57](#), [59](#), [60](#), [61](#), [62](#), [63](#), [64](#), [67](#), [68](#)

UML Unified Modeling Language. [46](#), [53](#)

VATic-TV Visual Attention Toolkit for TV. [60](#), [61](#), [64](#)

VR Virtual Reality. [18](#)

WAN Wide Area Network. [17](#)

XML Extensible Markup Language. [37](#), [38](#), [40](#), [53](#), [54](#), [55](#)

XSD XML Schema Definition. [53](#)

List of Figures

2.1	Quarterly Evolution of Rail Passenger Transport Performance at EU28 Level . . .	8
2.2	Cumulative Transport Infrastructure Investment to 2025	9
2.3	ERTMS Nano Simulator	10
2.4	ERTMS Compact Simulator	10
2.5	NSB Full-Cab Simulator	11
2.6	Cluster Architecture System	13
2.7	Example of Clusters System with Driving Desktop Simulator	14
2.8	Final design of the Cluster Sync Manager Interface	15
2.9	R ³ Database Synchronisation Status of Driving Desktop Simulator	16
2.10	Driving Desktop Simulator System	17
2.11	Infrabel 3D Cave Track Simulator	18
2.12	The Norwegian Train Crew Simulator	18
2.13	Stanford University's Driving Simulator	19
3.1	Representation of the Scanpath Over a Stimulus	23
3.2	TAPRAV Visualisation Tool Interface	25
3.3	Eye Movement Data Shown as Gaze Plot (a) and Space-Time Cube (b)	26
3.4	Fovio Eye Tracker (Engineered by Seeing Machines)	28
3.5	Changing Traffic Light Color Before Proposal Upgrade	30
3.6	Changing Traffic Light Color After Proposal Upgrade	30
3.7	Train Manager Before Proposal Upgrade	30
3.8	Train Manager After Proposal Upgrade	30
3.9	The Real Real Road After Proposal Upgrade	31
3.10	Usability Evaluation Protocol Schema	32
4.1	Usybus Architecture Example	34
4.2	Usybus Eye Tracking Monitor Interface	35
4.3	Usybus Pupils Sizes Monitor Interface	36
4.4	Usybus Eye Tracking Analyzer Interface	37
4.5	Usybus Messages Recorder Interface	38
4.6	Usybus Messages Player Interface	38
4.7	Usybus EyeWorks Eye-Tracker Controller Interface	39
4.8	Usybus Cognitive Load Monitor Interface	40
4.9	Usybus Task Scheduling Interface	41
4.10	Usybus Eye Tracking Monitor Controls After Upgrade	41
4.11	Usybus Eye Tracking Monitor Zone Filter	42
4.12	Usybus Agents Exchanging Message Interaction Sequence Diagram	47
4.13	Adaptive Interface as Usybus Agent Schema	48
4.14	The PDA-LPA Design Space	49
5.1	UsiTracker Component Toolbar	51
5.2	UsiTracker Main Window Interface	52

5.3	UsiTracker Components Dialogue Window	52
5.4	UsiTracker Menu ToolBar	53
5.5	Form Element - UML Class Schema	53
5.6	Grid Element - UML Class Schema	54
5.7	Wrapper Element - UML Class Schema	54
5.8	Constraint Element - UML Class Schema	55
5.9	Component Element - UML Class Schema	55
5.10	Param Element - UML Class Schema	56
5.11	Content Element - UML Class Schema	56
5.12	Tab Element - UML Class Schema	57
5.13	Border Element - UML Class Schema	57
5.14	Usybus Standard Widget Naming Dialogue	58
5.15	UsiTracker Settings Dialogue	60
5.16	User Interface Design Evaluation Cycle	62
5.17	Example of UsiTracker Eye Tracking Evaluation Environment	63
5.18	UsiTracker SmallApp User Interface	63
5.19	UsiTracker SmallApp VATic-TV Visual Measurements	64
5.20	UsiTracker SmallApp ICA Visual Measurements	64
5.21	UsiTracker SmallApp ICA Visual Measurements Boxplot	64
6.1	The Graph Analyser Sketch Specification	69
A.1	The Eye Tribe device	79
A.2	The Tobii 1750 device	80
A.3	The Tobii X50 device	80
A.4	The Tobii T60 device	81
A.5	The Tobii X60 device	82

List of Tables

2.1	Railway Passengers Transport in Million of Passenger-Kilometres	8
4.1	Usage Overview of the Usybus Types	46
5.1	UsiTracker SmallApp Instance Data Session	63

Chapter 1

Introduction

1.1 Context of the Problem

From the first board circuit to the apparition of computers, a lot of new technologies have emerged and surround us every day. The emergence of digital technology is nowadays indispensable for daily life. It is in this context of digitised era that researcher, computer scientist and company are always seeking for developing new innovative computer software skills and hardware technologies. Among those technical advances, computer simulation has emerged as one of the most widespread application uses in many domains. Whether this is for solving complex mathematical problem to the simulation of a tsunami, for flying simulators or even for simulation inside computer games, computer simulation has a large scale of utilisation. Simulation is jointly linked to the development of Information Technology (IT). Since the earliest use of computer simulation for the Manhattan project developed for the modelling of the nuclear detonation process, those disciplines have never ceased to grow together.

People are well aware of the stakes of computer simulation, so does the railway sector. Railway simulation has gone from strength to strength in recent years. Time and financial aspects are precious resources for railway company that have rail timetable schedules to respect while maintaining a quality transport service. The training of new drivers on the rail network could be dangerous and expensive in terms of equipment mobilisation and human resources. Train simulation is therefore a great alternative to reduce those risks and costs.

"Tell me and I forget, teach me and I may remember, involve me and I learn"

—Benjamin FRANKLIN

This quote from Benjamin FRANKLIN illustrates the state of minds of the railway simulation. The best way to learning to drive rolling stock is to combine theoretical aspect and procedural approach. The interaction of the trainee with real driving scenarios and the practice on the ground is the finest learning solutions for acquiring experience.

To support some of my dissertation arguments, Transurb Simulation the Belgian railway simulation company grants me the opportunity to pass their doors in the context of an internship. For some years now, Transurb works in conjunction with the Université catholique de Louvain (UCL) for different projects and frequently hire new trainees within the company.

1.2 Central Statement of the Dissertation

This master thesis will focus on eye tracking for designing the user interface of a multimodal train driving simulator and for evaluating its usage by learners. We will first try to understand the major issues of using simulation and more particularly for the train driving simulation context. As with all modern and new technological tools, railway simulations are in a perpetual process of development and engineers constantly attempt to make improvement of the actual state of the technology. In this dissertation we will discuss the different mechanics and methods of those progresses. Especially we will discuss the eye-tracking tool that allows capturing gaze points of a user. We will describe the technology and show its benefits when coupled to railway simulators, whether it is for interface evaluation or for simulation interaction.

To ensure the benefit of such a device, a small Human-Computer Interaction (HCI) research work as been made on eye tracking for the design of User Interface (UI) combined with the eye-tracking gaze plot of a user. The study with the help of the Usybus software environment shows the benefits of eye tracking over the User Interface development cycle and adaptive User Interface. Subsequently, we will explain how eye tracking can be paired with simulations, more particularly train simulation.

Finally, this dissertation will propose a concrete application of the eye-tracking tool developed on the simulator solution of Transurb which is the Graph Analyser. More than capturing point of gaze, the tracking device acquired by the UCL for research can capture the cognitive workload of its users that is a measurement of the mental activity of the subject. This information could help the instructor in his teaching tasks for rolling stock driving.

1.3 Coverage Assumptions and Concepts

The dissertation will help to familiarise the reader with railway simulations by covering large part of the solutions developed within Transurb Simulation and make him understand why simulation is used in such large scale in many domains. He will discover how the eye-tracking tools can improve the user experience. More particularly how it can enhance the human-machine interactions and improve the assessment of the perception of user actions in a virtual environment such as train driving simulation. Additionally, in this dissertation we will be focused solely on railway rolling stock, but sometimes we will reduce the name to "train".

1.4 Content of the Dissertation

The dissertation is structured in four distinct chapters.

- The first section covers more generally the simulation and especially railway simulation in a broader way. We will discuss more in detail the structure and the different software involved within Transurb Simulation company. Moreover, we will introduce you to the DB Sync Manager as a new module for simulator solutions within the company.
- In the second chapter we will discuss the eye-tracking technology and describe the actual state of the technology. We will discuss in a very wide range, the current applications of eye tracking and highlight some positive and negative features. Furthermore, we will also introduce you to the notion of cognitive workload. Finally, we will present a usability evaluation of simulation software from Transurb Simulation.

- The third division introduces the Usybus software environment. This agent-based architecture tries to bring more scalability and more convenience for eye-tracking systems within HCI search field. In this chapter we will describe the structure and the different agents of the Usybus. Moreover, we will show a test case of the Usybus tool by integrating eye tracking to adaptive user interface software.
- The fourth division presents the UsiTracker case study as a user interface editor tracker. The HCI search tries to establish the benefit of using eye tracking during the conception of UI. The visualisation of information (gaze point, fixations, cognitive workload) given by eye-tracking devices are useful to understand the user perception and user experience. The chapter will describe the UsiTracker tools used for the case study and present improvement of such a tool for User Interface evaluation.

Chapter 2

Train Simulation Overview

First of all, before going into the heart of the topic with the eye-tracking technology let's understand the basic of computer simulation. More precisely we will discuss the train driving simulation and the stakes that lies behind. We will focus on the Belgian company Transurb Simulation specialised in railway simulation by description of some real application of the railway driving simulation. We will end up by new technological breakthroughs and future development of this sector.

2.1 Computer Simulation

When you first hear about computer simulation, intuitively you think about complex physics systems representing the real world to simulate. You are not totally wrong since first simulations were used for that purpose. According to the Stanford Encyclopedia of Philosophy, computer simulation was pioneered as a scientific tool in meteorology and nuclear physics in the period directly following World War II [Win15]. More specifically it was the first large scale deployed during the Manhattan Project to model the process of nuclear detonation [KL13].

2.1.1 Formal Definition

A computer simulation is *a program that attempt to reproduce and simulate the behaviour of a particular system*. The representation of the system can be a mathematical description or an abstract model (a computer model). The system to simulate can be real as much as it can be an imaginary or hypothetical one. Thus, computer simulation field has a much wider arena than only physics or economic systems.

Following the definition given by Kim and Lee [KL13], "*simulation of a system is represented as the running of the system's model*". When running the computer model, the simulation calculated the system's state step by step. To achieve this, the algorithm takes in input the system's state at a defined time and calculate the system's state at an increased time and so on all along the simulation. As explained by Winsberg [Win15], "*the algorithm thus produces a numerical picture of the evolution of the system's state, as it is conceptualised in the model*". Therefore, according to Rajasekaran [Raj09], the validation of the simulation model is indispensable in order to assure the reliability and the trust of the simulation. Simulation can for example offer an overview into the development of a system that is too complex to simulate with analytical solutions. Moreover, in some cases the graphical interfaces allow the visualisation of the results of the calculations.

In a broader way, computer simulation is the process of designing a beforehand selected model. To implement that model in such a way that running the simulation is possible. To execute that algorithm and finally to extract and studying the outcome data resulting from the simulation.

2.1.2 Types of Computer Simulations

Computer simulation can be separated generally in two most common types. On one side, we have the agent-based simulation and on the other side equation-based simulation. In both cases we use a model to simulate the system. The former use an agent-based model consisted of multiple agents that encapsulate the behaviour of individuals. These individuals constitute an interaction network that forms the system being studied. The latter uses an equation-based model based on a set of differential equations that are evaluated during the simulation [VDPSR98].

2.1.3 Purposes of the Simulation

Simulation has several purposes and we will only enumerate the major ones. The underneath enumeration is therefore not exhaustive.

- First, simulation can predict the behaviour of some real system without observing it or recreating it in its real environment. Moreover, those simulation could easily change their system initial state. Imagine all the necessary efforts you would produce to observe or to force and study the collision of asteroids in real-life situation.
- Secondly, simulation grants the possibility to study systems or object behaviour in low-risk situations. Take the example of the studying of the nuclear detonation. Simulation reduces the risk for the whole process and avoid accident that could happen with a real test.
- Finally, simulation is a useful tool for learning process in the sense they can produce a life-like experience. Considering pilot learning in fly simulator. Simulation drastically reduces the risk of accident while learning, and can also place the pilots in different scenarios situation.

This last purpose introduce the **training simulation** notion. Most of the time those simulations are interactive. That means a human is included in the simulation process. The system is designed through a Human-in-the-Loop (HITL) simulation model. Most applications of training simulation are flight simulator or driving simulator and those simulators are commonly deployed in corporate situation.

We find multiple flight training simulations, like aircraft flight or helicopter flight simulator shipped by corporate like Thales Group¹ or SIMCOM Aviation Training² and many other companies. Driving training simulation has not been left behind. With L3 Training³ who proposes truck, bus, police car, fire truck, delivery van and snow plow simulator. In the same domain we found Transurb Simulation⁴ for railway driving simulator. We enumerated here only a fraction of existing solutions for training simulation. There exists even more company specialised in this domain we haven't mentioned.

¹Thales: Training & Simulation - <https://www.thalesgroup.com>

²SIMCOM Aviation Training - <http://www.simulator.com>

³L3 Training - <http://www.l-3training.com>

⁴Transurb Simulation - <http://simulation.transurb.com>

2.2 Understanding the Railway Simulation Motivations

Railway simulation has made a technological leap forward in recent years. With the growth of power processing, real-time applications like railway simulation is much more affordable. Performance in animated graphics and immersion elements like motion, sound and environment has enhanced and provides better experience in both flying and driving simulation area. Nowadays, railway companies cannot dispense with the railway simulation in order to prepare new drivers for their future job.

Indeed training new drivers on the network could be dangerous for the driver itself, but also for all rail network users. As said earlier, railway simulation eliminate those risks. The driver is immersed in an environment that can reproduce any driving situation without security issues. Failure and technical deficiency can be simulated without any exposure and material deterioration.

Moreover, the mobilisation of equipment and human resources can be minimised. Simulation is the most cost effective in driving learning. The installation is available and can be operational seven days a week. Simulation increases the productivity and save costs since it reduces the number of instructors required to train several students simultaneously.

Some other benefits are that rail simulation bypass the rolling stock monopolisation as well as the track occupation which avoids the network immobilisation for learning session. Moreover, railway simulation allows the addition of a multitude of events that can better prepare the driver. Like the simulation of technical problems, unexpected events, a large crowd of travellers.

Finally, in addition to instructors, the simulation learning process is associated with theoretical and procedural learning. Many interactive learning tools can be associated with driving which increases the trainee involvement. It allows the learner to gain an overall understanding of the railway simulation by making them familiar with the network, the signalisation, the radio communication protocol and more.

2.2.1 Railway Market Survey the Social and Economic Issues

Increasingly, there is a trend toward the use of driving simulators as a certification tool for training. The success of these simulators was most certainly favoured by the increasing congestion of the rails. Indeed, this trend will increase as a result of the continual increase in the number of passengers and goods transported by rail that cause unavailability of the tracks for ground training. The intensity of rail traffic also leads to the introduction of more specific safety and control systems that require targeted training.

Indeed, based on the "Railway passenger transport statistics - quarterly and annual data" survey⁵, this steady growth was observed between 2011 and 2015 in Europe. The Figure 2.1 illustrate the evolution of the rail passenger per quarterly for the 28 countries of the European Union. This evolution is measured in billion of passenger-kilometres (pkm) on the purple curve. The rail performance increased by 7.7 billion passenger-kilometres between 2014 and 2015 (+1.8 %). The tendency shows no sign of reversing with a total increase of approximately 36 billion passenger-kilometres between 2011 and 2015 (+8.49 %). Which gives us an average annual growth rate of +1.68 % per year. The graph also shows the growth rate for each year quarterly compared to the same quarterly of the previous year (the green sticks). Over those five years each quarterly allays recorded an increase compared to the previous rate. Although it was weak

⁵Eurostat - <http://ec.europa.eu/eurostat>

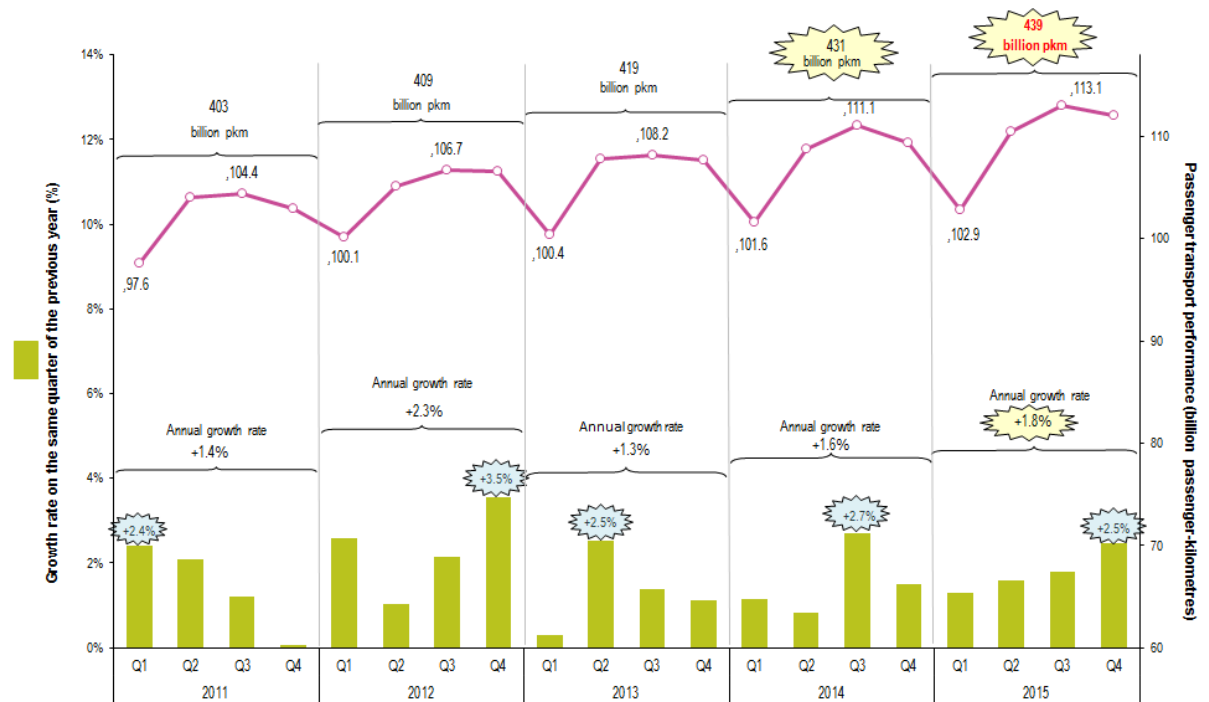


Figure 2.1: Quarterly Evolution of Rail Passenger Transport Performance at EU28 Level

Source: Eurostat [Eur17a]

for the fourth quarterly of 2011 and the first quarterly of 2013, other quarterly always had a relatively strong growth rate.

The Table 2.1 resume, in more details, the passengers transported for quarterly data only from major countries of the European Union. The unit of measure is also in millions passengers-kilometres. However, only the transport of passengers by train is considered. Metro, tram and light rail statistics are excluded. Those numbers represented in the table shows that the trend is conserved so far. There is an increasing passenger traffic density over the years and per quarterly to date.

	2014	2015				2016				2017
	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1
Germany	22,533	21,123	21,675	23,487	23,372	22,015	23,472	23,862	24,324	22,698
France	22,736	21,361	23,793	23,777	22,721	21,102	21,872	22,726	22,706	21,360
UK	16,773	16,043	16,525	17,043	16,983	16,576	16,915	17,292	17,228	16,969
Italy	12,119	12,019	13,536	12,718	12,884	12,492	13,118	12,728	13,107	12,565
Spain	6,331	5,879	6,604	6,390	6,604	6,172	6,866	6,804	6,690	6,384
Switzerland	4,821	4,591	4,754	4,837	4,914	4,738	4,899	4,918	5,033	4,808
Netherlands	5,336	4,238	4,427	4,163	4,695	4,352	4,541	4,278	4,809	4,466
Total	90,649	85,254	91,314	92,415	92,173	87,477	91,683	92,608	93,897	89,29

Table 2.1: Railway Passengers Transport in Million of Passenger-Kilometres

Source: Eurostat [Eur17b]

In contrast with other transport sector, the market of freight and passenger rail transport outlook to have a relatively strong growth. One of the reasons is the growing opinion in favour of

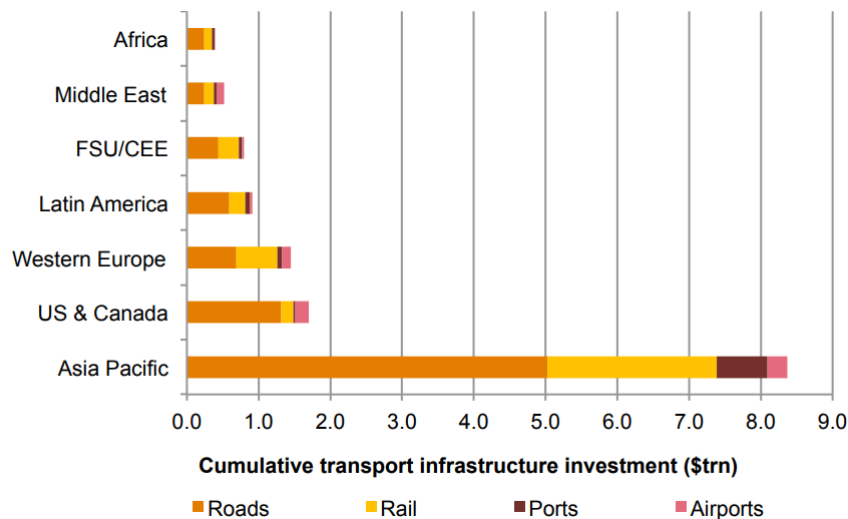


Figure 2.2: Cumulative Transport Infrastructure Investment to 2025

Source: [Oxford Economics](#) [HWS15]

public transport [HWS15]. Indeed, the forecast of transport infrastructure investment illustrated in Figure 2.2, shows higher investment in trillion dollars for rail transport just behind the roads compared to other individual subsectors.

2.3 Transurb Simulation for Train Simulator

This section aims to understand the different existing solutions for railway simulator by looking at the special case of Transurb Simulation company. We will also describe the structure and the utility of the different software within Transurb Simulation. Finally, we will discuss the creation of the DB SyncManager software that I developed during the two months of my internship at Transurb Simulation. This software is able to manage the synchronisation of the different databases of simulators stations within a local network. We will show the benefits of such an upgrade and how confronting data from different simulator devices can improve the user's learning and experience.

2.3.1 Solutions of Transurb Simulation

The principal products of Transurb Simulation are based around their simulators, the sections below describe the three different simulator solutions proposed by the company. The shipped simulators are principally for the simulation of train driving environment, but Transurb Simulation can also deliver simulators for metro or light rail (such as tramway) environments. Generally railway training simulator companies also have the same structured and merchandise three ranges of products as described in the following sections.

The Nano Simulator

The Nano Simulator illustrated in Figure 2.3 is a computer-based simulator. More easily transportable, the compact training unit consists of software and possibly a part of the command

console. Such a solution provides more convenient and flexible training sessions.



Figure 2.3: ERTMS⁶ Nano Simulator

Source: [Transurb Simulation](#) [Sim17c]

Indeed the Nano Simulator is more scalable and flexible because it is a less expensive and very lightweight simulator. The number of simultaneous training sessions can be increased and therefore, no time is lost with trainees waiting for a spot on a simulator. This solution is adapted to classroom training and designed for a first driving approach.

The Compact Simulator

The Compact Simulator illustrated in Figure 2.4 offers a wider immersion into the train driving environment. The simulator is delivered with a portion of the driving desk of the simulated vehicle that is easily switchable with other instrumentation panels. The Compact Simulator is well suited for training centres and takes really less space than the whole cab.



Figure 2.4: ERTMS Compact Simulator

Source: [Transurb Simulation](#) [Sim17a]

The Full-Cab Simulator

The Full-Cab simulator illustrated in Figure 2.5 is a one-to-one replica of the actual rolling wagon cabin simulated. The trainee can practise with real instrumentation in full immersion. The sound, the motion platform match the physics reaction of the simulated vehicle and an exact copy of the communication system contributes to this complete immersion.

⁶The European Railway Traffic Management System

⁷La Norges Statsbaner - Railway of the Norwegian State



Figure 2.5: NSB⁷ Full-Cab Simulator

Source: [Transurb Simulation](#) [Sim17b]

2.3.2 Existing software Modules

In order to review the SyncManager software, we need first to understand the general design of the simulators software within Transurb Simulation.

The simulation is made of different modules developed in C# under the integrated development environment tool Visual Studio⁸. Every simulator contains specific modules and a local or a remote database. The database store all information and data from simulator stations. More especially, it contains the different exercises and scenarios with their networks, signalisation, special events and theory learning. It also contains the user's information and the different reports of those users. Therefore, with remote database multiple simulator stations can connect to the same database server and share common information.

Each module has a specific role for the simulation process. In the sections below we will describe some of the principal modules and their functionality.

Process Manager Module

The Process Manager is a module dedicated to start and stop the simulation. More precisely, it's possible to configure the Process Manager to launch other modules in a desired order. The Process Manager is able to shut down one or all the working modules in a single action.

Sound Server Module

This module is useful for the sound of the simulation. The sound server manages the different sounds that can occur during the simulation. Such as the horn, clanging on the rails, the noise made by overhead line equipment and other sounds making simulation more realistic.

The 3D World (Unity)

This module is the core of the visual display of the simulation. Unity is a game engine able to create the 3D world of the simulation in real time. The platform is one of the most widespread for game and simulation graphics. The 3D world module is therefore the interface toward which the driver will observe the real-like environment of the simulation.

⁸IDE Visual Studio - <https://www.visualstudio.com>

Virtual Train Module

The Virtual Train is a module that allows you to control the train by displaying the different instruments. Instead of having a physical command panel, virtual driving desk for example can be used in Nano Simulator . The module also contains a main view of the simulated rolling stock. This view allows a trainee to access the relevant compartments of the vehicle in the simulation in order to execute extra actions on panels that are not located in the driving cab.

Real Rail Road Module (R³)

The Real Rail Road (R³) software is the principal tool of Transurb for the simulation. This software is installed on each simulator and manages the simulation sessions. Administrators and instructors are able to create new simulation scenarios and exercises by choosing a network and adding a series of events. It's possible in R³ to manage a track network by creating and adding signalisation. Trainees can launch simulation by running a scenario. R³ software displays a report at the end of the simulation exercise in order to give feedback to the trainee.

TransNet Module and TcpBridge Module

Modules are independent software and are useless if they do not interact with each other. However, this is possible through the TransNet module. TransNet is the communication server between the different simulations software. Data are transmitted through this server and an interface allows the user to see and manage the connection status of the different modules. While the TcpBridge is a specific module that only links the TransNet module to the Unity 3D World module. Thanks to its own protocols, TcpBridge acts like a bridge for transmitting data from TransNet to Unity and conversely.

Track Editor Tool

The Track Editor tool is not strictly speaking a module. Indeed, this software is used to create track network, but is not used during a simulation. The created network, with the help of an earth view, is saved inside the database of the simulator.

2.3.3 Driving Desktop Simulator System

The notion of Cluster and Driving Desktop Simulator is important to define and to understand before describing the DB SyncManager Module illustrated in the Section [2.3.4](#).

Driving Desktop Simulator

Driving Desktop Simulator (DDS) is corresponding to a driving simulator station. This simulator station must have a local database. The reason of that specification is justified by the fact that DDS should be able to work on stand-alone mode. Meaning that the DDS can fully work on its own like any simulator solutions provided by Transurb Simulation.

Cluster

A cluster is an association of several DDSs managed by at least one instructor station. The instructor station is a particular post that controls the simulation of the different trainees. The connection with the instructor station is made through a wireless local area network, more precisely the local Wi-Fi network. To form a cluster, instructors just need to authorise desktop simulators to get connected on their Wi-Fi network.

Moreover, database of DDSs are all synchronised within the cluster. Each DDS can be easily added or removed from a cluster, so the DDS can easily move from a cluster to another one. The DDS, whether or not belonging to a cluster, that is not connected to the Wi-Fi network (not connected to the cluster) is considered being in stand-alone mode. Whenever the DDS that belong to a cluster reconnect to the Wi-Fi network (the DDS was in stand-alone mode), the database is synchronised with the cluster.

Finally, the network of a cluster is secured with a firewall. The Figure 2.6 illustrates the cluster architecture system. In the figure the database of the instructor station is local, but since instructor stations do not necessarily have to work in stand-alone mode, we could have a remote database on another computer.

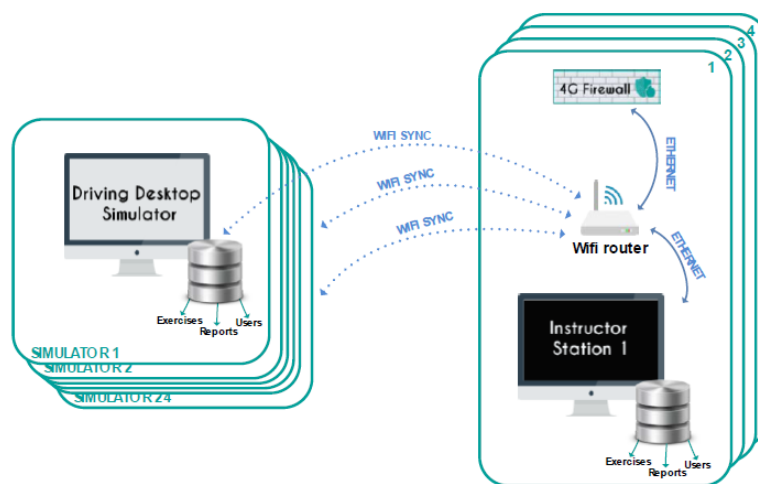


Figure 2.6: Cluster Architecture System

2.3.4 DB SyncManager Module

The DB SyncManager module is the new module implemented in the context of my internship within Transurb Simulation during two months⁹. As well as the modules described previously in Section 2.3.2, the DB SyncManager module also has a well-defined role. This new module, with the help of the Cluster Sync Manager application, is able to manage clusters in the simplest way possible. The application also manages automatically the database synchronisation between DDSs within the cluster. The DB SyncManager software can be launched in two distinct modes.

- **Instructor mode:** this mode is activated on instructor station. That means it's an administrator mode able to manage the cluster system by adding or removing DDSs in a

⁹The Scrum methodology was applied during the software development.

single click. Cluster Sync Manager launched in instructor mode is, therefore, the database synchronisation server.

- **Trainee mode:** this mode is simply a DDS station where learner drivers can train and practise on the simulator. It acts as a client of the cluster manager server. Trainee mode has been integrated to R³ software. There are two major reasons for that feature.

First, it doesn't make sense to have a second program running on simulators station when R³ is already installed on each simulator and needs to be launched for the simulation.

Secondly, having a user interface only for the instructor mode is easier to configure and is less ambiguous to use. Moreover, the DB SyncManager module running in instructor mode can be launched on any station with a remote or a local database and acting as a server. While the DDS in trainee mode needs to be used with a simulator station with local database so it makes possible the stand-alone utilisation of the simulator.

Example of Clustering System Configuration

The Figure 2.7 illustrates an example of a Clusters System with associated Driving Desktop Simulator of the developed solution. The rectangles define the stations while the two rounded black frames represent the range of the Wi-Fi Network signal. While dotted lines within areas represent the UDP¹⁰ protocol communication connections between DDSs and Cluster Sync Managers.

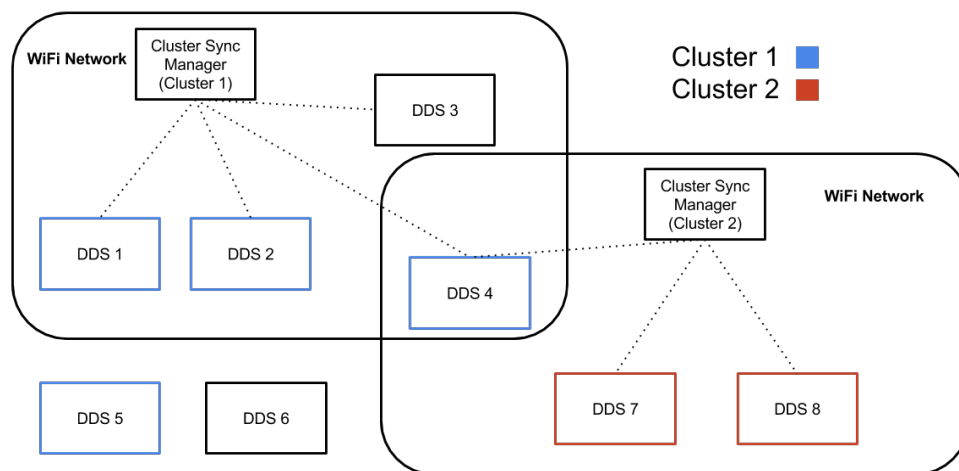


Figure 2.7: Example of Clusters System with Driving Desktop Simulator

There are two clusters illustrated in Figure 2.7, the **cluster 1** in blue and the **cluster 2** in red, each one managed by the respective Cluster Sync Manager.

- The **cluster 1** is composed of 4 DDSs, the DDS1, the DDS2 the DDS4 and the DDS5 while **cluster 2** is composed of DDS7 and DDS8.
- The DDS5 is not within the range of the Wi-Fi of the **cluster 1** or is simply disconnected from the Wi-Fi. This DDS still belongs to **cluster 1** and acts as a stand-alone simulator that will be synchronised once it reconnects to the Wi-Fi network.

¹⁰The User Datagram Protocol (UDP) is an internet communication protocol using connectionless transmission of packets

- The DDS6 and DDS3 are simply simulators with no associated cluster, but the DDS3 is visible by the Manager controlling the **cluster 1**, thus, it can be added to this cluster.
- Finally, the DDS4 is the only Driving Desktop Simulator visible by both cluster Manager, this DDS belonging to **cluster 1** can easily change to **cluster 2**. Note that changing DDS from one cluster to another erases his local database before synchronising with the new associated cluster.

2.3.5 SyncManager user interface

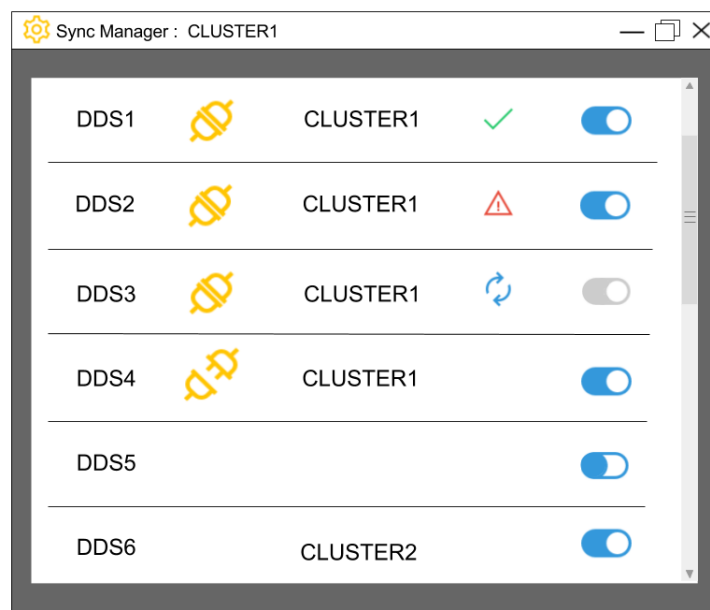


Figure 2.8: Final design of the Cluster Sync Manager Interface

The Figure 2.8 illustrates the final user interface of the SyncManager¹¹ in instructor mode. In this figure, the Cluster Sync Manager that manages **CLUSTER1** (see the application header) has a list of Driving Desktop Simulators (the first column) with their associated cluster (the third column). Empty cluster name means no associated cluster for the corresponding DDS. Therefore, its local database is not synchronised at all in any cluster. The DDS behave like standard simulator solutions, previously described in Section 2.3.1, in stand-alone mode.

The plug symbol represents the mode of the DDS.

-  If it is plugged, the DDS is connected to the Wi-Fi network and synchronisation with the cluster is possible.
-  While if it is unplugged, the associated DDS within the cluster is not connected to the Wi-Fi network. The DDS is therefore working in stand-alone mode.

DDS with no plug symbol are simply the DDS visible on the network, but that does not belong to our cluster. Moreover, just on the right of the associated cluster we have the symbols for the synchronisation status that are respectively from top to bottom:

¹¹The .NET application is using Window Presentation Foundation (WPF) for user interaction with the methodology Model-View-ViewModel (MVVM) as architecture formalism.

-  **Synchronisation successful:** the last synchronisation of the database with the corresponding DDS ended well. Both instructor and trainee are up-to-date.
-  **Synchronisation failed:** the last synchronisation failed for some reason. This could be due to connectivity reason, database version, etc.
-  **Synchronising:** the databases of the instructor and trainee stations are synchronising.

Finally, the blue toggle button is simply a switch to add the DDS within the cluster or to remove the DDS from its associated cluster. You can notice that the button cannot be switched when the DDS is synchronising. However it's also the case when the DDS is in simulation. Moreover, if the database version differs from the cluster manager, the DDS cannot be added to the cluster.

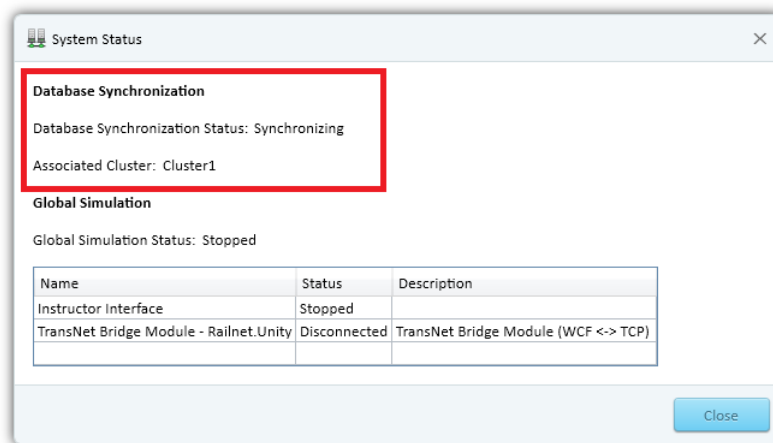


Figure 2.9: R³ Database Synchronisation Status of Driving Desktop Simulator

The Figure 2.9 illustrates the system status pop-up window of the R³ software. This window originally displayed only the global status of the simulation. The status of the database synchronisation and the associated cluster of the DDS has been added as shown within the red square. When the DDS is on stand-alone mode, the database synchronisation status is replaced by the mention "Standalone".

2.3.6 DB SyncManager Intent and Enhancement

The primary goal of the development of that new module is to design a scalable and flexible training classroom. Indeed, with the last solution, two choices were possible to set up simulators for training room.

- Each simulator has his own database, but synchronisation was never performed between them. Therefore, exercises, user data and reports were only accessible locally on the simulator device.
- Simulators have a central remote database (for example the database of the instructor station) where they all share the same information. However, stand-alone mode for simulator is impossible since they need to stay connected to that remote database.

The DB SyncManager Module provides a solution to those two limitations. Training session can be organised by forming clusters. A classroom is therefore composed of a number of DDS simulators and instructor stations. Size of the classroom is adjustable at convenience. Simulator can be added or removed to the next room or training session easily and quickly just by adding or removing this DDS from the cluster. This solution is optimised for Nano Simulator as illustrated in Figure 2.10. The figure shows Driving Desktop Simulator System as scalable classroom with three DDSs and one instructor station. This system solve the current driving simulator disadvantages which is simulator systems require lots of infrastructure and do not allow the training of a large number of students in parallel.



Figure 2.10: Driving Desktop Simulator System

Moreover, the DB SynManager module is the starting point tool to wider database synchronisation. While the databases are all synchronised in a Local Area Network (LAN) at this time, future development could lead to synchronisation on Wide Area Network (WAN) by considering the synchronised cluster LAN network as a single database. Creation of exercises, creation of network rail with signalisation and report consultation could be decentralised.

Indeed, the study conducted in 2008 by 2Train based on a sample of 118 simulators marketed at 17 railway operators [SM08], states that 20% of the simulators systems can be accessed through remote connection, but it usually remains only for maintenance. Furthermore, the study mentions that 90% of simulators are distributed applications that are running on more than one computer. The DDS System innovation try to invert this tendency with lightweight simulators running only on one single computer in remote network.

2.3.7 Advanced Training Solutions

Transurb Simulation also proposes advanced training solutions in the railway expertise. We will highlight the 3D Cave Track Simulator and the Train Crew Simulator respectively illustrated with pictures at Figure 2.11 and 2.12. Those diversified solutions fall under the scope of the training simulation. CAVE technology also know as the cave automatic virtual environment can project in three dimensions the simulation environment. The participant is therefore immersed in that simulated world where he can practise procedures for railway network maintenance. In the case of the track simulator, track worker can practise without traffic exposure and related

security issues. The Train Crew Simulator is more a full-scale test of the train crew procedure.



Figure 2.11: Infrabel 3D Cave Track Simulator

Source: [Transurb Simulation \[Sim17e\]](#)



Figure 2.12: The Norwegian Train Crew Simulator

Source: [Transurb Simulation \[Sim17d\]](#)

More and more training simulation tends to modernise and take the opportunity to adapt with new technology. Training simulation should be immersive, interactive and informative to give the best learning condition and experiences.

2.4 Training Simulation Future Development

Future of the simulation is paired with technological breakthroughs. Many innovative tools have come up such as virtual reality headset, Google glass, hand tracker and finger tracker, or eye tracker. Those tools provide new way of thinking in training simulation. Although Virtual Reality (VR) was at an experimental stage two decades ago, the technology is currently far more developed. Simulation is already very active in military or medical assessment. Those two domains are exploiting the full potential of the VR technology. Virtual training simulation can train new recruits or prepare to situations they never faced, but might happen in their career.

Great examples are VR parachuting simulation or wearable VR system to train troops tactics and teamwork in combat situations. The VR coupled with flight training for aircraft and helicopter or driving training for any sort of vehicle provides better immersion than ever. We also find in medicine VR surgery training [DLBR97] or VR radiographic learning for unlimited training where mistakes are free of consequences.

Moreover, the momentum given by the VR brings new innovations as the fusion between real world (reality) and the virtual world (models). This combination of both real world and VR is called the Augmented Reality (AR). While VR is already popular, utilisation of the AR in simulation is an increasingly widespread phenomenon. Users within real-environment can interact with virtual object to give a unique HCI experiences [CWC14]. As mentioned by Che et al. [CLH11] it is now possible in AR to incorporate geographic element and capture evolution of spatial objects in real time that could be used for designing simulation environments. In a near future we could envisage having AR glasses to overlay on equipment like command panels, some information to help the driver. Such technology could be a real-time assistance on rolling stock, but also in full replica railway simulator.

Among all those technologies, eye-tracking system also has a great future in training simulation. Eye Tracking is already present at this time in driving simulator. Stanford University has created a car driving simulator coupled with eye tracking and facial expression analysis. In this simulator, gaze points can be captured by using eye tracking glasses while a camera analyses the facial expression.



Figure 2.13: Stanford University's Driving Simulator

These days, railway simulation is not commonly equipped with such device. Like the utilisation of eye tracking in car driving simulator, railway driving simulator could also be equipped with eye-tracking technology. Training simulator needs proper assessment and feedback in order to optimally improve the skills of the learner. Those evaluations are enhanced with this technology since it provides feedback on how visual attention is distributed, this has never considered before.

Source: [iMotions](#) [iMo17a]

Chapter 3

Eye-tracking an Emerging Technology

The eye tracking is *the measurement of eye activity, more specifically the study of the eye and the ocular behaviour*. In this chapter we will describe this innovative process in fairly extensive way. Although eye tracking opens up a wide area of use, we will focus mainly on the scope of eye tracking for user evaluation and for HCI domain. Eye tracking data is collected using either a remote or head-mounted device called the eye tracker or the oculometer. This emerging and evolving technology is a new process of data capture. Eye tracking is adding as a complement to the existing interactive data measurement from instruments devices like the keyboard, the microphone, the mouse, the pad, the joystick, the camera and more recently the touch screen.

3.1 Eye Tracking Terminology

In this section we will define the different metrics relative to eye tracking. Most are derived from the eye gaze, but as you will see some are related to pupils size. First of all, oculometer are collecting gaze points with a certain frequency. This frequency varies in function of the type of devices and can go from 30 Hz up to 1250 Hz¹. However, usually oculometers operate more around 50 Hz or 60 Hz. Thus, the frequency of the oculometer indicates how many gaze points per second are recorded. Oculometers use infra-red camera to detect the pupil-iris boundary and the corneal reflection. Depending on where the corneal reflection position compared to the pupil the oculometer is able to calculate gaze points. However, raw data as eye gaze points do not provide a general overview of eye movement. Rather, eye tracking use different data types that will be described here.

Fixations

Fixations occur when the vision is focused on relativity small area for a certain time. The eye gaze, thus, forms an aggregation of points. The range of fixation duration is between 150 ms and 600 ms [Duc07]. Fixation take most of the viewing time and this is where the brain can process visual information. Fixation is represented by a cycloid area. The centroid of the fixation can be measured in pixels with the x-axes and y-axes coordinate. The fixation diameter is proportional to the fixation duration or the dispersion depending on the algorithm that measure fixation and is generally around 20 to 50 pixels.

¹According to Wikipedia - [Eye Tracking](#) [Wik17a]

Saccades

Saccades are fast eye movements toward a new location of the visual environment. Those extremely rapid eye movements jumps from one fixation to another. Typically the range in duration of saccades goes from 10 ms to 100 ms [Duc07]. This short duration renders the executor effectively blind, thus visual information is suppressed during the transition.

Smooth Pursuits

Smooth Pursuits occur when eyes, by matching the velocity, are tracking a moving target. The stability of the visual image is preserved during the pursuit, but it depends on the range of the target motion. During the smooth pursuit, any other objects next to the target are badly processed. The angular velocity limit is around 30°s^{-1} , over that limit, eye movements aren't able to perform smooth pursuits, but instead we will observe sporadic saccades².

Stimulus

Eye-tacking Stimulus can be seen as the visual content. This stimulus can be static or dynamic and are presented to participants during an eye-tracking experiment. Moreover, the content of the stimulus can be either active or passive. Generally, stimulus are 2D contents, however eye tracking tends to have more and more a 3D approach.

Zones of Interests

Zones of Interests or Areas of Interests define parts of the stimulus that are significantly important. Therefore, those subregions of the displayed stimuli, are highlighted since there are important for the experiment. Data can thus be extracted specifically from these regions. The zone of interest could also be a model or object in the case of a 3D stimulus.

Scanpaths

Scanpaths are sequences of consecutive fixations and saccades. Scanpaths are therefore traces through time and space. It gives information about the scanning strategy of a participant [GH10]. Commonly, fixations of scanpaths are represented with ellipses, whereas saccades are characterised by a straight line. Scanpaths of participants are recorded over stimulus (the space) during a certain amount of time.

The Figure 3.1 illustrates a scanpath over a stimulus. We can observe the different gaze points, fixations and saccades, but also the zones of interests that have been defined. The numbers are the logical order of appearance of alternating fixations and saccades that constitute the scanpath. In this representation, the stimulus is in-plane. The scanpath can overlap itself, this generally occurs when the eye movement over a stimulus is chaotic.

²According to Wikipedia - [Smooth Pursuit](#) [Wik17b]

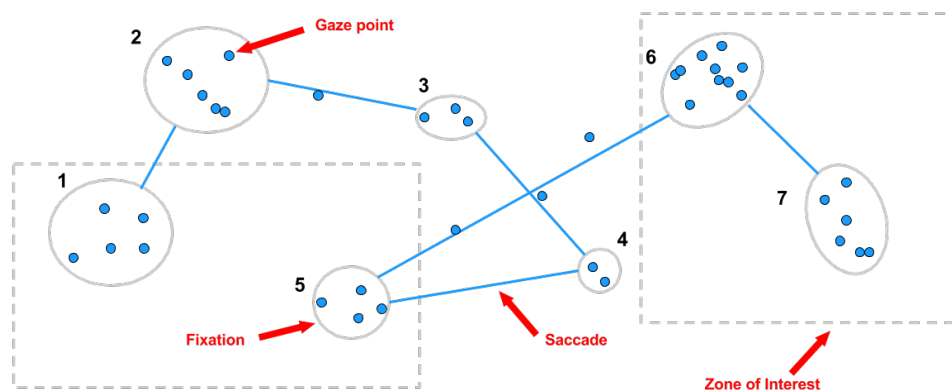


Figure 3.1: Representation of the Scanpath Over a Stimulus

Index of Cognitive Activity

The Index of Cognitive Activity (ICA) is a metric derived from the cognitive workload. The cognitive workload is the measurable mental effort provided by an individual that has been assigned one or more cognitive tasks. The cognitive workload (or mental workload) can be measured in three different ways, by performances for a given task, physiologically (pupil dilation, heart rate, etc.) or subjectively.

The ICA measures the cognitive workload with the changes in pupil dilation (physiologically measure). More specifically, ICA is calculated during a continuous recording of pupil diameter by measuring the number of abrupt discontinuities per second [Mar02]. Moreover, ICA makes the distinction between pupils changes due to light reflex and dilation reflex because of the presence of cognitive efforts. The ICA range varies from 0 to 1. The higher the ICA value, the higher cognitive workload and the higher the chance that the observed person is in difficulty, overwhelmed by a certain task.

3.2 State of the Art of Current Applications

A significant number of papers and research studies exist on the field nowadays. This increase is a direct result of the easier and cheaper use of eye-tracking technology. In this section we will have an overview the different applications of eye-tracking system. The enumeration is not exhaustive, but tries the cover as much as possible the large field of the eye-tracking sphere.

3.2.1 Eye Tracking for User Evaluation

This first category regroups eye-tracking applications that are said diagnostic. Programmer aims to extract the user's visual and attentional patterns over a given stimulus. This can be for example, which areas someone is viewed in a photo or ad space. Those zones of interest do not sincerely move or react to the user's gaze. The eye tracker in diagnostic application is generally unobtrusive, so subjects are not aware of its presence. Otherwise, if the user is aware of its presence, it could change his behaviour.

Psychology and Neuroscience Domain

The domain regroups the eye tracking research applications to understand the causal links between what we are watching and how we react with the information collecting with our eyes.

Visual attention can be correlated with other measurements in order to detect evidence of unsuspected pathology. Indeed, some researches in experimental psychology and clinical neuroscience have demonstrated a link between mental disorder and the eye movement. As explained by Vidal et al. [VTBG12], portable eye-tracking systems can help detect cognitive disorders and illnesses like Alzheimer's disease, autism, dyslexia, drug consumption, schizophrenia, multiple sclerosis and the HIV infection complication named AIDS Dementia Complex.

Industrial Engineering and Human Factors

Concerning scene perception and simulation, the task is to study how humans behave physically and psychologically in comparison with particular environments. We already mentioned in the previous chapter the utilisation of eye tracking for driving training simulation with the Stanford University's study. Still in driving simulator, a study from Palink et al. [PKSH10] is evaluating the cognitive workload and the performance of the driver. However, eye tracking can also be applied in real driving situation [KWKJ15]. Another example is the exploration of urban surroundings with KIBITZER a wearable gaze-sensitive system [BFH10].

Eye tracking has a role to play in the study of the awareness behaviour and the visual spatial attention. Indeed, the eye might fix a certain location, but the spatial attention can be directed to other locations [MF01]. The scanpath over a visual stimulus might not completely represent the reality of regions fully perceived. Koivisto et al. [KHR04] show how irrelevant stimulus might affect our spatial attention. Eye tracking is used for the assessment of situational awareness in order to reduce human factors in marine [YWLB13] and flight operations [YWLB14].

Marketing and Advertising Analysis

Companies nowadays track more and more user consumer habits for the purpose of targeting advertisements to users on the basis of observed or known personal characteristics. Commercial websites are taking advantage of eye tracking in order to determine the interest of the user by tracking not only mouse position, actions and user shopping cart, but also the assessment of visual attention [AWS06]. Page web designers should be aware of the eye tracking for website optimisation and usability [PHG⁺04], [CBD02].

There is a need in marketing to determine while looking at advertisements what our attention is focused on [HLR14] or if consumers are looking at advertising banners on websites [RA14]. Moreover, the eye-tracking marketing and consumer search field also include packaging design and shopper behaviour in shopping stores [Cle07].

Evaluation of User Experience

Eye tracking is utilised for evaluating the usability and can be applied for measuring user experience. We already mention its utilisation for web pages, but eye tracking is still in search for other usability evaluation software. The challenge stands in the difficulty to find appropriate ways to use and interpret the data [PB05]. Goldberg [GK99] introduced how eye tracking improve

computer interface evaluations between poor and good graphical interface design.

Concerning form interfaces, the eye tracking study helped to demonstrate that forms are filled quicker if labels are near the fields and above the boxes [Pen06]. According to Galesic et al. [GTCC08] users spend more time looking at the first options rather than those at the end of a questionnaire list. While performing a usability eye tracking test, Romano and Chen [RC11] demonstrate the importance of where error messages should appear.

Eye Tracking Data Visualisation

All the previous examples use continuous, visual and physiological measures. Visualisation is therefore important in order to understand and rapidly explore relationships among the data. Therefore, Bailey et al. [BBI07] introduced TAPRAV illustrated in Figure 3.2 that provides interactive task model and pupil response curve visualisation.

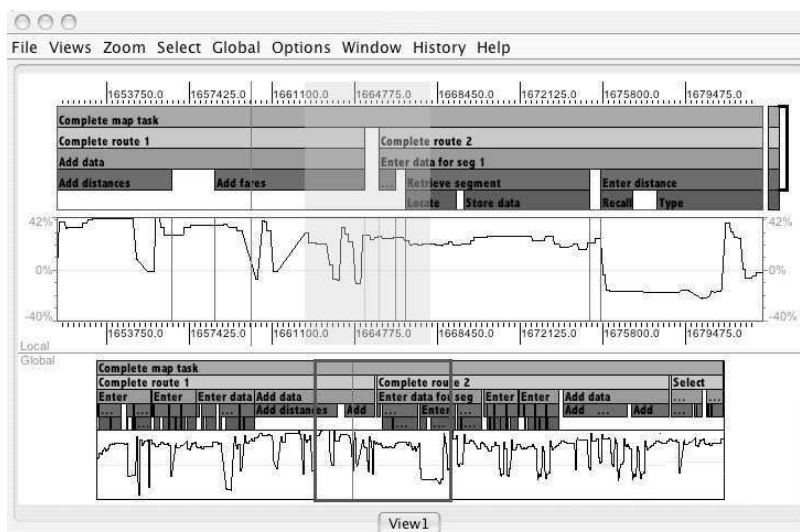


Figure 3.2: TAPRAV Visualisation Tool Interface

Source : University of Illinois at Urbana-Champaign [BBI07]

Goldberg and Helfman [GH10] introduced how to extend the scanpath representations with different scanning strategies. The authors highlight the different representations advantages and in which case it can be used. Heat-map is well appreciated as a data visualisation technique. Extended heat-map visualisation is proposed, where eye gaze data add opacity or transparency to the background image [ŠM07].

Li et al. [LÇK10] and Kurzhals et al. [KW13] proposed respectively 3D space-time visualisation for gaze plot over static and dynamic 2D plane stimulus. The former combines typical eye movement analysis methods and space-time cube frequently used in geovisualisation as illustrated in Figure 3.3. The latter also propose space-time cube visualisation, but adjusts fixation data based on the moving object via clustering computation.

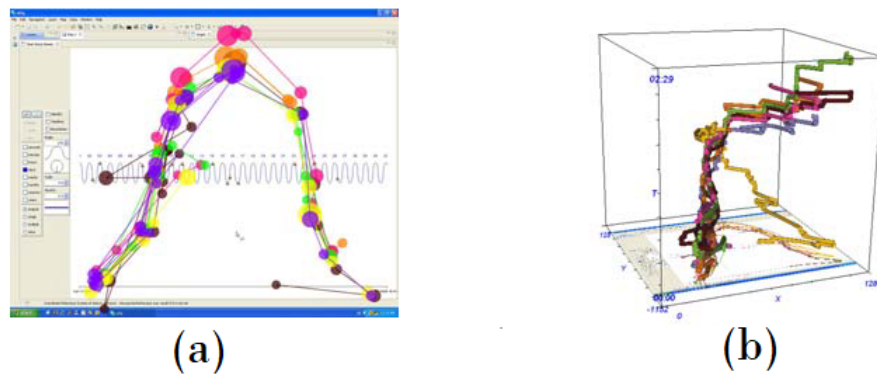


Figure 3.3: Eye Movement Data Shown as Gaze Plot (a) and Space-Time Cube (b)

Source : ITC-University of Twente [LÇK10]

3.2.2 Eye Tracking for User Interaction

The second category regroups interactive applications. Interactive systems are expected to respond to the user's gaze in one way or another. In other words, the user is fully aware of the presence of the eye tracker and can use it to interact with the machine.

Eye Tracking for Human-Computer Interaction

Eye-based interactive systems can interact with the machine using gaze as a pointing device. The advantage resides in the fact that users hands might be free for other tasks and eye-gaze input system is faster than selection with a mouse [SJ00], [Mur06]. In the field of usability, [BI02] Bates and Istance compared performances between traditional eye mouse and a zoom screen facility on standard graphical user interface. Moreover, Drewes1 and Schmidt [DS07] proposed sequence gaze gesture to prevent unintentional eye selections instead of delaying the response time.

In the context of driving assistance, eye tracking is used as real time eye detection for drowsiness and fatigue of automobile drivers. The non-intrusive device works for variable and realistic lighting conditions which normally affect the performances of eye tracking [GNM15]. Moreover, the eye gaze can be utilised as a command interface to control drones [HAMM14].

Eye Control for People With Disabilities

A great example in eye tracking is the utilisation of eye gaze as input for people with motor disabilities. Hybrid system using electromyogram³ signals from face muscles and eye-gaze tracking have been implemented as a hands-free assistive technology [CBCA08].

Gaming Eye Interactions

Eye tracking also has applications in the gaming industry. Corcoran et al. [CNPB12] have developed real-time eye tracking and face tracking using only user-facing camera. Such system has a wide range of potential applications in gaming, but could also be applied to smartphones and tablets for new perspective of interactivity.

³Device to measures the electrical activity produced by muscles.

Tobii Gaming⁴ already proposes tracking devices and software compatible with a large range of popular games. As a complement keyboard and mouse, eye tracking can adjust the viewing angle of the in-game camera, but it can also be used to react in some elements of the game or blend control panels not watched by the gamer.

3.3 State of the Art Analyses

We will first analyse the existing solutions and discuss the strengths and weaknesses of eye tracking for railway model applications. These analyses will aim to put into perspective the need for UsiTracker to use the eye tracking Usybus architecture. The case study analysis aims to provide a better environment for HCI eye-tracking software and to reduce the HCI conception and evaluation cycle.

First of all, there is no current study in the domain of eye tracking for railways, nowadays the attention is focused on car driving. Only a few studies can be found concerning training simulations, the rest are in general more based on eye tracking in Human-Computer Interaction and Usability research in a broader way. Researchers are still trying to figure out how to correctly interpret all that quantitative visual data. Moreover, there are some limitations to eye tracking, the first one is that you can't tell why a viewer is looking at something. Additionally, we can't certainly say that a user consciously sees something or not, respectively due to lack of awareness or due to spatial attention somewhere else of the gaze point.

Just as the utilisation of eye tracking for driving or flight simulators, the capture of eye movement could also be applied in railway training simulator. Three improvements can be highlighted by integrating eye tracking to railway simulation.

- The first benefit would be the diagnostic approach concerning human factors in railway environment. Collecting eye tracking from full-cab simulation environment while operating rolling stock could identify behaviour of drivers. The goal is to reduce human error and increase the safety by observing how operator are responding to some events. In the same domain, operator needs constantly to perform safety-related and communication tasks while operating the train. Eye tracking could be a tool to optimise location of control panels and equipment. Finding design flaws and deficiencies by testing railway rolling stock environment on simulators before producing the locomotive would be more cost-efficient.
- A second advantage would be the utilisation of tracking eye movement for real-time interaction with the simulation, but also for post-simulation feedback. In the first case, it could be used as a learning tool. The simulator could interact with the trainee when he is looking at some areas or if missed important elements like for example a limit-speed signalisation. This mode with embedded eye tracker could be easily activated or disable for more realism and immersion. In the second case, measurement as the cognitive workload could provide feedback on the difficulty experienced by the train driver during given training task.
- Finally, eye tracking could be useful for simulator software usability evaluation. That means the evaluation of the user experience and the evaluation of the user interface together. Software should be easy to use and intuitive whether it is utilised by trainees or instructors.

Secondly, despite the fact that a lot of eye-tracking visualisation tools emerged such as TAPRAV, 2D visualisation, none or only a few of them propose real-time data analysis. Visualisation is most of the time performed post-experiences. Moreover, there is no common standard

⁴Tobii Gaming - <https://tobiigaming.com>

toolkit for data visualisation and data analysis. Companies propose their own solutions and software, while researchers always build new environment for their specific eye-tracking domain of research. Hence there is a need for a scalable tool and an infrastructure independent of the oculometer for the integration of HCIs in eye-tracking environment.

Finally, concerning eye tracking for user experience and UI evaluation, case studies always focus on the final software. Most of the time eye-tracking evaluation is always performed on a specific interface after its conception, but never during the development stage. Incorporating eye-tracking evaluation during the User Interface design cycle could enhance the search field for UI development.

3.4 EyeWorks and the Fovio Eye Tracker

Given the growing interest in eye tracking, several oculometers have been proposed for research initially, but now also as a consumer-oriented product. Popular ones are Tobii⁵, iMotions⁶, Gazepoint⁷, TheEyeTribe⁸ (before it was acquired by Oculus VR), Pupil Labs⁹, LC Technologies¹⁰, EyeTech¹¹ and finally EyeTracking¹²



Figure 3.4: Fovio Eye Tracker (Engineered by Seeing Machines)

Source: [Tea Ergo](#) (dispenser)

Among all that choice, we opted for the FOVIO oculometer from EyeTracking (see Figure 3.4), a non-invasive stereoscopic binocular system (two cameras) allowing measurement in 3D at a rate of 60 Hz. The FOVIO can record the head movements of the subject, the direction of the gaze, the eye movement (saccades, pupillary diameter, eye vergence), but also the monitoring of lip and eyebrow movements.

The software from EyeTracking that is delivered with the FOVIO is EyeWorks eye-tracking software. Nevertheless, this software is also compatible with some oculometers from EyeTech, Tobii, LC Technologies and others. EyeWorks is used to step-up the scene environment, to configure testing script of the experiment, to perform the calibration of the oculometer and to record the collection of data. Moreover, EyeWorks contains some tools in order to analyse a recorded experiment.

⁵Tobii - <https://www.tobii.com>

⁶iMotions - <https://imotions.com>

⁷Gazepoint - <https://www.gazept.com>

⁸TheEyeTribe - www.theeyetribe.com

⁹Pupil Labs - <https://pupil-labs.com>

¹⁰LC Technologies - www.eyegaze.com

¹¹EyeTech - <http://www.eyetechds.com>

¹²EyeTracking - <http://www.eyetracking.com>

We chose FOVIO because this eye tracker has some specific characteristics that other oculometers do not have.

- Firstly, among the different records information we listed above, the FOVIO is the only oculometer that is recording mental workload by providing the Index of Cognitive Activity standard. Other devices are compatible¹³ to provide the cognitive workload indicator only under the utilisation of EyeWorks. Indeed, EyeTracking have a patent concerning his search in the measure of the ICA [Mar08], and therefore it is the only one that can provide that measure.
- The second specificity comes with his Multi-Display module. When all oculometers only work for single screen display, the FOVIO can record across multiple screens simultaneously. Additionally, the multiple displays do not necessarily have to be coplanar and do not necessarily need to be linked to the monitoring computer.
- Finally, the third particularity, with the FOVIO it is feasible to track against physical objects with the Scene Camera Module. Therefore, it is possible to configure multiple independent systems into a single multi-camera system providing merged metrics.

In the context of railway simulation, the operator cab involves several zones like the area of vision through the windows, the lateral zones, the controls, the sector of the communicating system. Those zones are located in different planes and we need a large tracking range. Most current eye trackers offer only one recording area at a time. When they offer several, they impose that the surfaces are in the same plane, which is not the case with the FOVIO device.

3.5 Usability Evaluation of Real Rail Road

To put in test the FOVIO, a usability evaluation of R³ software from Transurb (described in Section 2.3.2) have been performed under the direction of Jorge Luis PEREZ MEDINA. The study was part of the Transurb Simulation TrainSIM project in collaboration with the UCL. The main goal was to identify elements of the UI design that were problematic. Meaning elements that would lead to ineffective and unsatisfying experiences for the user when performing certain given tasks. The final goal was to suggest recommendations for this software to improve the overall productivity and reduce frustration of the users.

The interface of R³ evaluated during this experiment was divided into three different sections:

1. The train section to manage trains on the network for the simulation.
2. The lookup section that shows aerial map view of the simulation network with the different element that make-up that network.
3. The simulation event section to manage the different events occurring during the stimulation like signal changes, weather changes, tool-tip pop-up window, etc.

Besides the visual measures from eye tracking, a survey has been presented to the participants in order to obtain additional qualitative data for the analysis. Additionally, video and audio were recorded as supplementary data for the evaluation. The participants had to perform four tasks on R³ in ascending order of difficulty while narrating what they were thinking and doing. The evaluation pointed out some usability problems and some improvements have been

¹³For the compatibility list see <http://www.eyetracking.com/Hardware/Eye-Trackers>

proposed. Most of the time, those proposals were uncovered by the feedbacks and qualitative data. However, eye-tracking data has revealed anomalies in the design of the software due to an overload or a lack of fixations in certain area of interest when participants encountered difficulties.

Among those difficulties, participants were disrupted and were not sure that the task was done when changing the signal states of lighting signalisation on the network. Indeed, the traffic light did not visually change colour when its status was changed. The proposal made to overcome this shortcoming was:

Proposal: To solve this problem, a solution would be to associate the change of the traffic light with the change of its colour on the map.

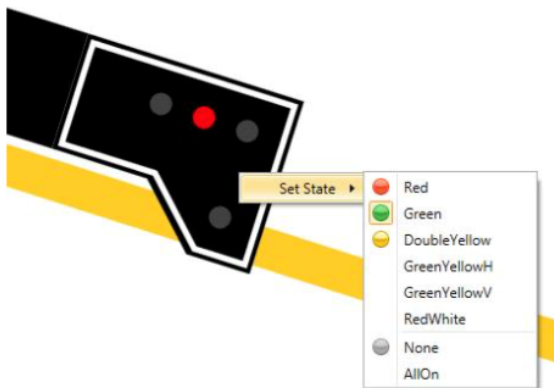


Figure 3.5: Changing Traffic Light Color Before Proposal Upgrade

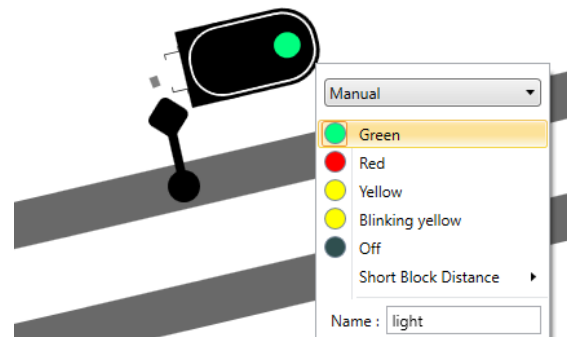


Figure 3.6: Changing Traffic Light Color After Proposal Upgrade

Additionally, many users do not even see the button to add a train. This button was the "Train Library" button. Even those who saw it did not necessarily understand its usefulness. The proposal made to overcome this shortcoming was:

Proposal: We can add a button "Add Train" and then only, offer the choice of the type of trains within the library. This button should be next to the train representation.



Figure 3.7: Train Manager Before Proposal Upgrade

With "Train Libray" Button (in red frame)

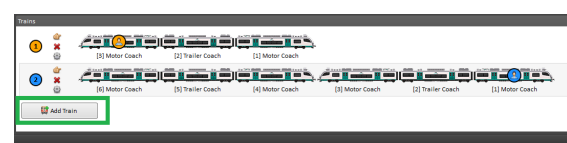


Figure 3.8: Train Manager After Proposal Upgrade

With "Add Train" Button (in green frame)

Moreover, some participants had trouble with the navigation on the areal map. Participants had to click on vertical or horizontal scrollbar or use and click on the scroll button for navigation. The interface of the map goes against their first intuition to use left or right click to navigate and the scrolling button for zooming. The proposal made to overcome this shortcoming was:

Proposal: The software should change the interface so that the user can move more intuitively by maps dragging. Another idea is to add a small thumbnail map in the lower right corner with the navigation keys as Google Maps does. The ideal here would be to be able to combine these two ideas.

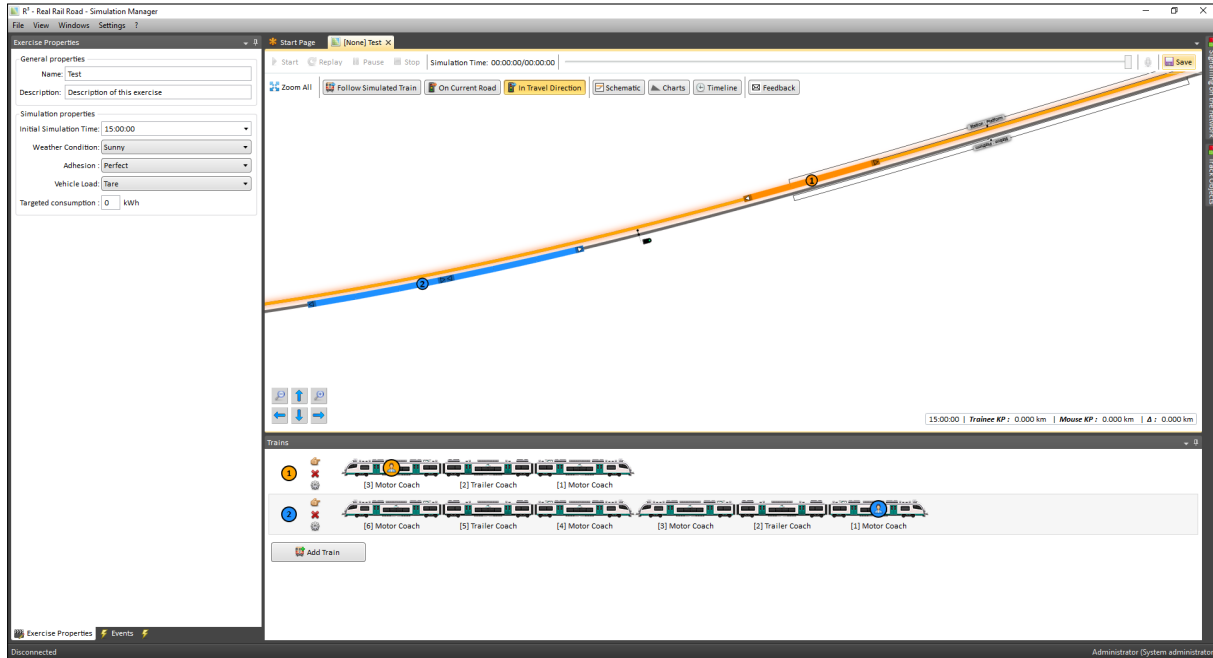


Figure 3.9: The Real Real Road After Proposal Upgrade

The Figure 3.9 illustrates the R^3 with the three distinct sections of the interface that have been evaluated. This figure shows the R^3 software after the usability evaluation. More particularly for the areal network view, we can see there is no more scroll bar and navigation keys have been added in the bottom left corner of the map. The navigation is possible with the right click of the mouse. Moreover, the button "Add Train" have been added under the list of trains.

3.5.1 Procedure to Conduct an Eye Tracking HCI Evaluation Experiment

The experimental protocol for usability evaluation consists of three phases as illustrated in Figure 3.10.

1. During the **Prepare** phase the supervisor of the usability testing will identify the task for the experiment. The supervisor will prepare the basis and the different scripts for the experiment. In this case the EyeWorks software has been used to prepare a set of instructions that will be presented to the participants during the session. Moreover the supervisor needs to set-up and install all necessary tools for the tracking experiment. The supervisor will also evaluate the number of test participants needed (between 5 and 39 [PN09]). Afterwards, he will contact the different participants and brief them on the objectives they will have to accomplish. If needed, a consent form has to be signed so the participants are aware of the terms and conditions of the experiment.
2. During the **Collect** phase, participants will, one by one, take place in the testing environment for the experiment. In order to collect visual measurement, a calibration needs to be

performed for each user before starting the session. The calibration¹⁴ will ensure quality in precision of the measurements. In this case the calibration is done with EyeWorks software before launching the scripts. Once the calibration is done, user can start executing the tasks they were assigned. Each user's visual attention is recorded, and video or sound track of the session could also be recorded. At the end of the session the participant may be invited to fill a survey form.

3. During the **Analyse** all data collected from participants will be used to identify elements of the application that are or that could be problematic. During the analysis, collected data from participants need to be compared and classified to generate the output of relevant characteristics. In this case the EyeWorks allows the statistics computation and visualisation of the data. However that process is not automatic, participants' data have to be imported, zones of interest have to be defined manually, etc.

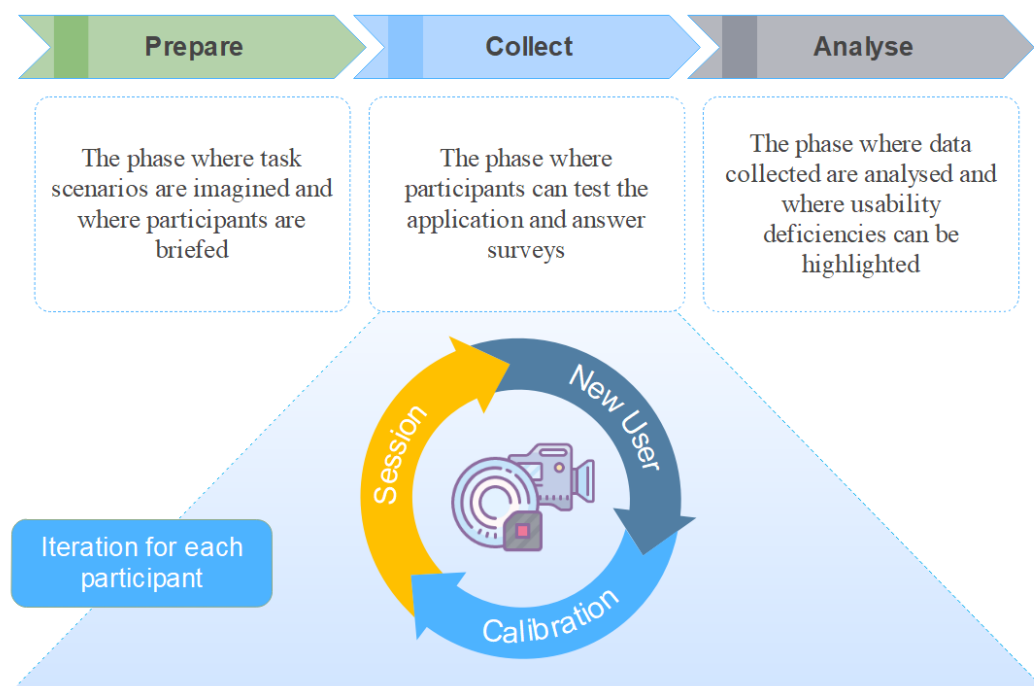


Figure 3.10: Usability Evaluation Protocol Schema

¹⁴By following a point on the screen, the eye tracker can measure the characteristics of the user's eyes.

Chapter 4

Usybus Software Environment for HCI Study

In this chapter we will describe the software environment set-up for the case study we perform in the continuity of HCI usability evaluation. More particularly we will discuss the multi-agent architecture Usybus. The case study aims to understand how eye-tracking system could enhance the interaction between the machine and the human user. The study has been separated in two parts that are both concerning the HCI field.

- The first and smallest part (described in this chapter) is concerning the utilisation of eye tracking coupled with adaptive user interfaces. In this situation we are using the oculometer in order to detect a person's visual patterns. Those patterns can show the interest of the user and can be used as feedback to trigger adaptation of the interface.
- The second part will focus on the UsiTracker software which is an interface editor application that allows a user to create his UI easily. This software is including eye tracking for interface design analysis and further search in HCI domain. By integrating eye tracking directly into the design phase we try to reduce design-evaluation cycle of the conception of UIs.

The final goal of the search is the integration of that technology in a near future for railway training simulation interface designing and for multimodal train simulator by creating visualisation interactions between the trainee and the simulator. Meaning that at the same time the trainee will be able to operate his controls and receive a physical, a haptic feedback (through the simulator screen, the motion or the sound) and, in the meantime, we will capture his facial expression, his eye gaze, his cognitive activity and his behaviour making it possible to give him advice.

4.1 Usybus Multi Agents Architecture

In order to conduct eye-tracking experiments we need some tools to be able to capture the data from the oculometer. We already mentioned earlier the EyeWorks desktop application from EyeTracking. This software is a good tool to collect data from oculometers supported by EyeWorks. However it has some drawback when performing experiment for HCI usability and design evaluation.

- First, we are not free to use the oculometer we want, although EyeWorks is compatible with other devices this is not the existing majority. Moreover, the installations and the configurations to conduct an eye-tracking experiment applied to HCI with EyeWorks software are sometimes really fastidious.

- Secondly, the software does not perform real time data capture analyses from the oculometer. It only records data and video session for post-experiment analyses. Moreover, each area of interest needs to be defined manually after the session ended. This process will become laborious, especially in the context of HCI usability evaluation experiment, if each widget or portion of the UI needs its own zone of interest. Additionally, those defined zones are static which is a real problem if the user interface is dynamic.

Furthermore, in the context of the train driving simulation, the instructor should be able to observe in real time what information are retrieved from the trainee. More specifically, to see in real time the gaze trace, the fixations in zones of interest and the cognitive workload during a simulation. One of the main reasons for such a functionality resides in the fact that the instructor could react quickly if the trainee has some difficulties.

This is where we introduce you to the Usybus Agents to overcome those shortcomings. Developed by Francis JAMBON from the University of Grenoble Alpes in the Grenoble Informatics Laboratory, the Usybus is an oculometric data processing system designed on a multi-agent architecture [JL12]. Agents can communicate through a bus of data. This bus called Usybus (for Usability Data Bus) is based on the open-source Ivy bus library software¹. The Ivy library allows applications to exchange information, it is a broadcast channel where agents send messages and other agents subscribe to the reception of certain messages. Ivy is designed in such a way to implement light interactive systems and powerful generic HCIs software [BBC⁺02]. Therefore, Usybus aims at simplifying the communication between devices in usability tests. It also enables flexibility and versatility of the Ivy platform.

On the Usybus, every information is transmitted in a format close to JSON with an associated timecode, to uniquely define each data and the order in which they were produced. Agents connected to the Usybus have well-defined roles, they can emit and retrieve data on the Usybus data bus without carrying about other agents. Agent using data on the bus, do not need to know which agent sent the information. Conversely sending agent does not need to know which agent will use their data. More precisely we have for example an agent that retrieves gross oculometric data, another one for displaying data, another one for sending the positions of the zones of interests, another one for the calculation of the gaze plot and much more relevant agents.

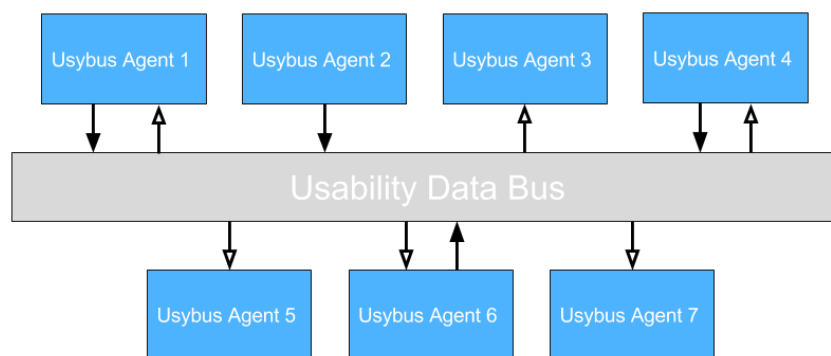


Figure 4.1: Usybus Architecture Example

The Figure 4.1 represent an example of a simple version of the Usybus architecture. In grey we have the Usability Data Bus with the flow of transiting data. In blue we have different agents connected to the Usybus. The agents transmitting data to the bus are represented by the arrows

¹Ivy library - <http://www.eei.cena.fr/products/ivy>

toward the data bus, while listening and collecting data agents are represented by the incoming arrows toward the agent. Naturally, as shown in the figure agents can endorse both roles.

4.2 Existing Usybus Agents

In the sections below we will describe the different existing agents and their respective functionality. Each agent is an executable JAR file implemented in Java. Therefore, Usybus agent can be run on any Operating System (OS) for which the Java Virtual Machine (JVM) is installed. Generally, agents can be started and stopped emitting or receiving data on the Usybus at any time thanks to a "Start" and a "Stop" button.

Moreover, because every sending agents have an associated device name, each receiving agent can filter and receive data only from a specified device. This functionality allows us to have multiple agents launched and connected to one Usybus, but running only for their associated devices.

4.2.1 The Usybus Eye Tracking Monitor

This agent is one of the main agents of the Usybus architecture. His role is to display in real time the gaze points and the fixations, but also the zones of interests. The agent interface is illustrated in Figure 4.2. The Usybus Eye Tracking monitor is thus a listening agent on the bus and displays data from any device or for the selected device. Moreover, the interface also displays the name and the time in milliseconds of the zones fixed by the user. Those fixations in zones are useful if we want to exactly know which zones a user fixed in a certain past amount of time.

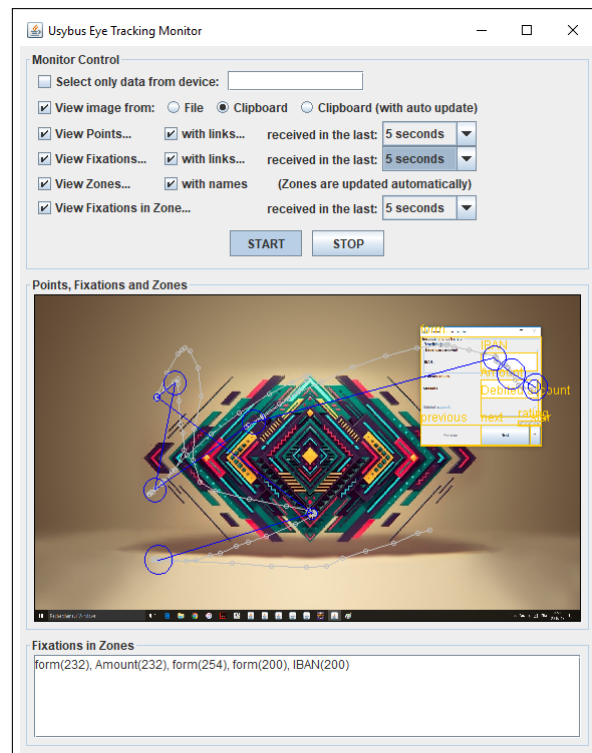


Figure 4.2: Usybus Eye Tracking Monitor Interface

The user is free to select which data he wants to display thanks to the checkboxes. It is possible to draw or not the links between gaze points and fixation center point. Furthermore, the observed data are displayed for a certain amount of time that can be specified by the user. Finally, the monitor can also display a view image (like the devices display screen) from a file or from the clipboard (like the print-screen). The clipboard with auto update radio button automatically update the view image if a new image is set in the clipboard.

4.2.2 The Usybus Pupils Sizes Monitor

The Pupils Sizes Monitor agent is a simple agent displaying the pupils size in millimetres for both eyes. The interface shows two blue circles representing the pupils size and their diameter value in real time. The agent display a red cross when there is no data available. That means there is no data on the bus or the data might be of too low quality, but also when a user blinks his eyes. The Figure 4.3 illustrates the interface of the agent. Even though there is no history in time, it can be interesting to observe in real time the size of the pupils. Indeed, this agent is a first step toward the analysis of the cognitive workload.

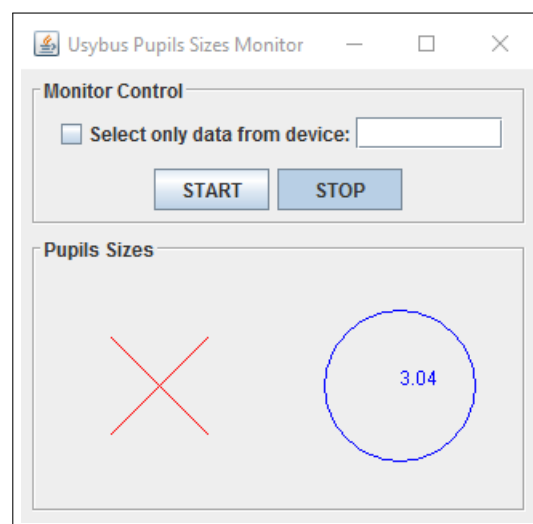


Figure 4.3: Usybus Pupils Sizes Monitor Interface

4.2.3 The Usybus Eye Tracking Analyzer

The Usybus Eye Tracking Analyzer illustrated in Figure 4.4 is an interesting agent. Indeed, this agent is able to capture data concerning eye gaze points and return to the bus new data concerning the eye fixations. The analysis can be performed with several algorithms that are:

- **Widdel algorithm:** The original Widdel algorithm published by Salvucci under the name Dispersion-Threshold Identification (I-DT). The algorithm that is one of the most used, identifies fixations as groups of consecutive points within a particular dispersion [SG00].
- **Sliding Window algorithm:** An experimental algorithm based on a sliding window that is still under research and development by Francis JAMBON from Laboratoire d'Informatique de Grenoble (LIG).
- **Minimum Fixation algorithm:** An algorithm also under investigation by LIG. The algorithm is based on a search for minimum fixing size.

- **Overlapping Fixations algorithm:** The algorithm makes possible to detect the beginnings of fixations also called "premises of fixations". In contrary with other fixation detection that perform an analysis after the acquisition phase. Overlapping Fixations uses a method called "on-line" that makes it possible to detect the fixation in real time that has just started. The approach of the algorithm is linked to the I-DT fixation algorithm [Jam13].

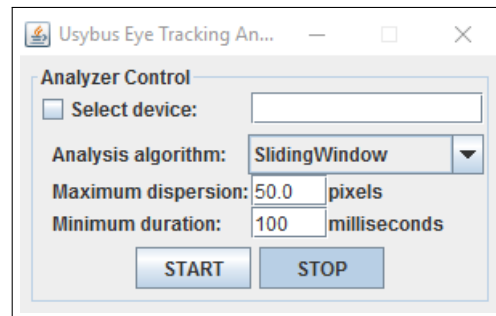


Figure 4.4: Usybus Eye Tracking Analyzer Interface

Additionally, the agent is capable of receiving the zones of interest to calculate and return the fixations in zones. This means; which zones as been fixed and for which amount of time the fixation last.

4.2.4 The Usybus Controller Agents

Controller agents are the agents collecting the data from the oculometer. They act as a bridge between the eye tracker device and the Usybus data bus. There are two controllers, **TheEyeTribe Controller** and the **Tobii Eye-Tracker Controller**.

The former is a controller dedicated to the oculometer of The Eye Tribe². The hardware specification of this device is defined in the Appendix A.1. The interface allows you to connect and retrieve the status of the eye tracker like the connection state, the calibration state, the frame rate and the average calibration error. The agent allows you to choose which data from the oculometer to send in real time to the Usybus.

The latter is a controller agent for Tobii³ eye trackers. This agent is able to connect to different Tobii eye trackers to extract data received from the oculometers. The implementation is based on the Tobii SDK2 software. Therefore, the agent is compatible with all Tobii "50 Series". The compatible devices are specified in the Appendix A.2, those oculometers are mainly dedicated to the research.

Moreover, this agent has also advanced options like the "Monitor" option to verify if the eye-tracking works properly or like the "Calibration" option to calibrate the eye tracker for the user eyes without having another interface to do the task. Finally, the agent also contains a clock synchronisation indicator. This indicator indicates the drift between clocks of the eye tracker and the computer system clock.

²Eye Tribe Eyetracking - <http://theeyetribe.com>

³Tobii Eyetracking - <https://www.tobii.com>

4.2.5 The Usybus Messages Recorder / Player

Those two agents, as their names explicitly imply can record data and replay recorded data on the Usybus. The achievement of those agents is to be able to save an experiment and replay its events at the time as they occurred, as many times as necessary. The messages are recorded under the Data folder to Extensible Markup Language (XML) files. On the Figure 4.5 we can see the interface of the Usybus Message Recorder. It is possible to specify the XML file name. The agent needs to know which type of data it should record, this list of types can be edited manually or be provided with a text file or the agent can listen himself on the Usybus and remember all the types of data which passes through the bus. The description of the different type of data can be found in the Section 4.6. The agent is able to start and stop recording whenever we want. As other listening agents, it can filter to record only the data from the selected device.

The player agent in Figure 4.6 is able to read the XML format of the file created by the recorder agent and to replay the sequences of messages on the Usybus. Some options, like the repeat option, allow us to repeat the sequences when it ends, but also send original timecode or filter messages with the selected device name.

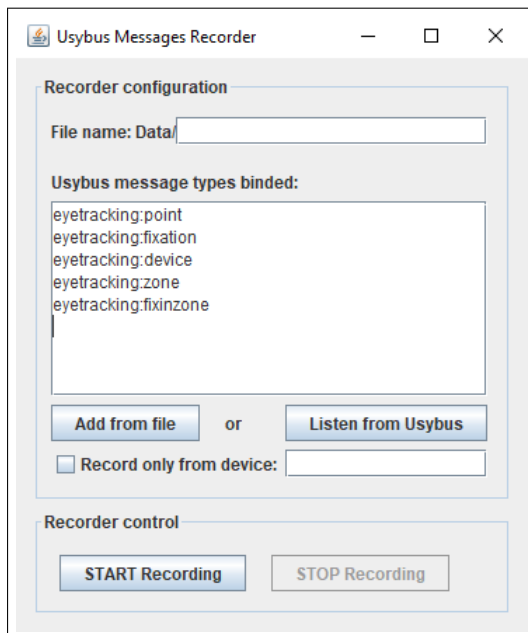


Figure 4.5: Usybus Messages Recorder Interface

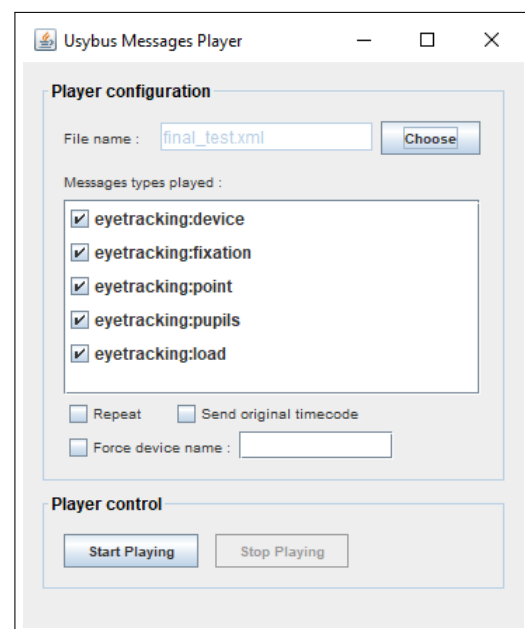


Figure 4.6: Usybus Messages Player Interface

4.3 Usybus EyeWorks Controller agent

The Usybus EyeWorks Controller agent is a new implemented agent. As the previous controller agents, this agent is collecting data from the oculometer. Both existing controller agents (TheEyeTribe and the Tobii Eye-Tracker Controller agents) are not compatible with EyeWorks eye tracker devices. The Usybus EyeWorks Controller agent is therefore necessary if we want to use devices from EyeWorks Inc.

The interface of the agent is illustrated in Figure 4.7. This agent allows, thanks the two top buttons "Connect and "Disconnect", to connect or to disconnect the EyeWorks Data Server. The EyeWorks data server is the software that makes the link between the eye tracker device

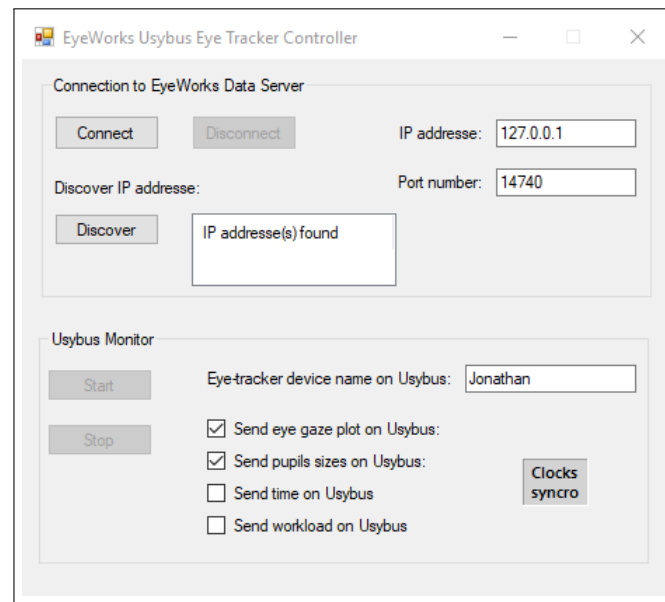


Figure 4.7: Usybus EyeWorks Eye-Tracker Controller Interface

(connected in USB) and the programs that use it. The server is located at an IP address for a certain port that must be specified in order to establish the connection. For convenience, the button "Discover" allows the user to locate the IP address of all EyeWorks instance running on the local network.

Once the controller agent is connected to the data server, we can manage and choose data that is sent on the Usybus thanks to the Monitor section. When the button "Start" is pressed, any data coming from the EyeWorks server (any data from the oculometer) are sent on the Usybus if the data checkbox is selected. Moreover, like any data emitting agent on the Usybus, the device name must be specified. Finally, you can notice that the agent also contains a clock synchronisation color status. The clock indicates the drift between clock of the oculometer and the computer system clock's as the reference. There are three possible warning color status:

- **Green indicator:** the drift is equal or below 100 milliseconds. The drift is negligible.
- **Orange indicator:** the drift is higher than 100 milliseconds, but equal or below 1000 milliseconds. The drift is acceptable.
- **Red indicator:** the drift is higher than 1000 milliseconds. We consider that the drift is too high and no longer acceptable.

When the drift is too high, the data that can be found on the Usybus do not represent any more data captured by the eye tracker in real time. This delay can be caused by many reasons. Eye tracking software needs lots of computational power when running. The computer can run out of memory resources that cause the desynchronising of clocks. However let's remember that drift depends on the clock's quality and the environment of the device, thus the rate could be different between the two clocks.

The source code of the agents is implemented in C# with the help of the AppConnect SDK for developers provided from EyeWorks. The SDK allows us to easily retrieve data from the oculometer with callback functions that are triggered for any new incoming data. You would have noticed that this agent is able to send cognitive workload on the Usybus. This feature as mentioned in Section 3.4 is specific for some eye tracker devices compatible with EyeWorks.

Therefore, thank to the SDK, it is possible to retrieve the left and right Index of Cognitive Activity as an indicator of the cognitive workload. Thus, the need of a new agent to visualise data from cognitive activity in real time.

4.4 Usybus Cognitive Workload Monitor Agent

To meet the needs of the representation of the cognitive workload a new agent has been implemented. This agent allows the user to observe in real time the evolution of the left and right Index of Cognitive Activity thanks to a line chart. In the case of the railway simulation, instructors could analyse during a simulation if a trainee as a higher cognitive activity in comparison with a cognitive activity reference like an experienced driver. The Figure 4.8 illustrates the interface of the agent.

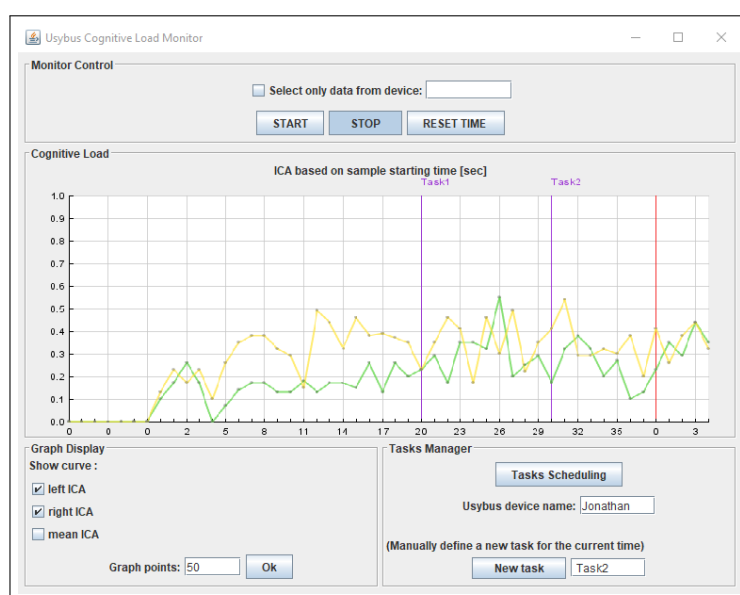


Figure 4.8: Usybus Cognitive Load Monitor Interface

The Usybus Cognitive Load Monitor is able to listen to the Usybus and retrieve data concerning the cognitive activity from the oculometer. Like other receiving agents, it is possible to filter data only for a specified device. The graph display the Index of Cognitive Activity as a function of time. The y-axis of the ICA measurement varies between 0 (the lowest activity) and 1 the highest activity measurement. The x-axis is the measurement of the experiment time in seconds since the first data received. The button "Reset Time" set the current time to a new zero time reference point. This new reference time is shown by the vertical red line on the graph. Furthermore, it's possible to display the mean between the left and the right ICA as well to change chart scale by increasing or reducing the number of graph points displayed.

The agent allows users to define tasks for an eye tracking running session. Defining task is important for the cognitive activity analysis since tasks could have different realisation solutions and-or different levels of difficulty. Starting a new task can be done manually by writing the task name and clicking on the button "New task". New tasks are set for the current time and are shown by a purple vertical line on the graph. The task name is writing on the top of this line. The observer can thus know exactly when tasks started and which portion of the ICA chart correspond to the associated task.

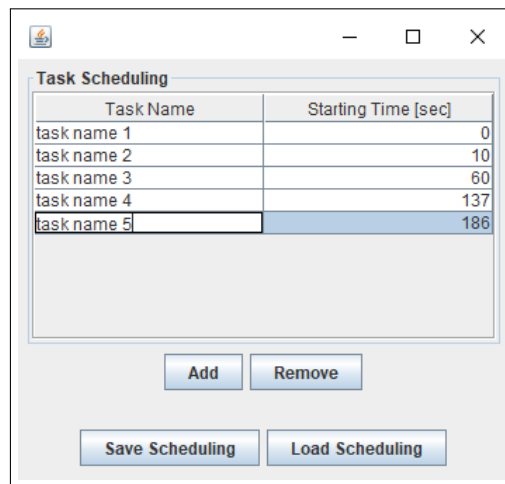


Figure 4.9: Usybus Task Scheduling Interface

Additionally, it is possible to draw up a time schedule for the monitoring tasks. The interface is illustrated in the Figure 4.9 and can be opened with the button "Task Scheduling" from the Cognitive Load Monitor. The table with the task name and the starting time in seconds, relatively to the zero reference time, can be completed in order to organise task scheduling. It is possible to add or easily remove lines of the scheduling table. Once a schedule is made, it can be saved in XML format file and can be loaded back for further use. Tasks are sent on the Usybus, therefore any other agent could potentially use this data.

4.5 Upgrade of the Usybus Eye Tracking Monitor Agent

Few upgrades have been made to the original Usybus Eye Tracking Monitor agent. Those upgrades enhance the feedback analysis of a session. This first change is the capacity of saving a picture of the monitor display. The capture, with the "Print" button can be saved in low or high resolution in the specified folder as a PNG image. This capture incorporates everything that is displayed on the monitor at the time the button was pressed. That means the view image, but also the gaze points, the fixations and the zones. Furthermore, when the "Auto print" checkbox is selected, a new capture is saved each time the monitor display a new view image.

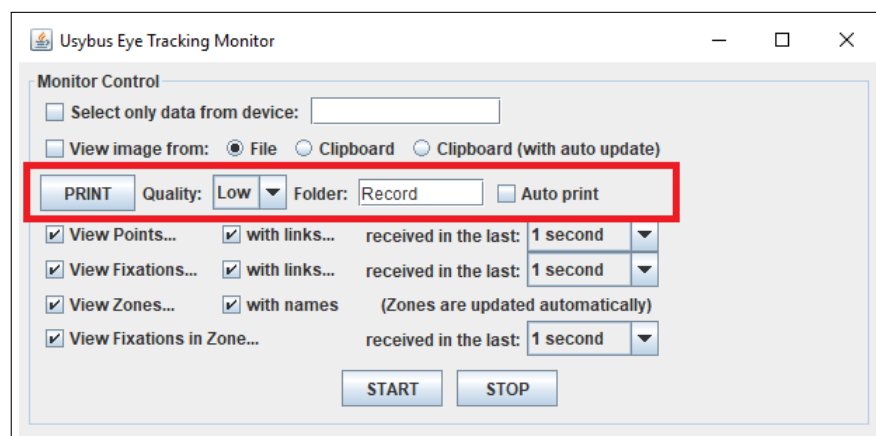


Figure 4.10: Usybus Eye Tracking Monitor Controls After Upgrade

The second upgrade concerns the zones. A filter has been implemented in order to select and display only the zones of specific Java **Swing**⁴ components. The interface is illustrated in the Figure 4.11 and shows the list of all **Swing** widget elements that can be filtered. This window can be opened with a new button from the Usybus Monitor called "Zone". In our case study this feature will help us to focus on the results of a certain zone or a combination of zones. This filter is allowed to show only the zones that really interest us. By default all components are selected, therefore all **Swing** widget components are displayed.

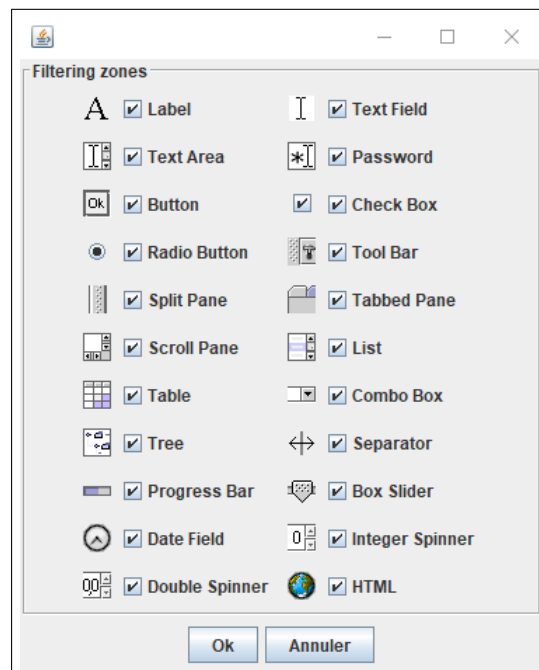


Figure 4.11: Usybus Eye Tracking Monitor Zone Filter

4.6 Usybus data types

Every data on the Usybus are typed messages, thus receiving agent only subscribe to data types they are interested in. The architecture of the agents is not defined statically but dynamically according to the types of data issued and subscriptions to these types. Moreover, the notion of devices, we talked earlier, makes possible if needed, to distinguish different transmitters providing the same type of data. On the following sections we will describe which agents are able to generate such messages and for each data type we will define its structure.

Type Point

This type is describing the eye-gaze point relatively to the device for a defined timecode. It is the gaze coordinates in pixels on the screen where the user's eyes are looking. The type also has an optional information telling if yes or not the gaze point is fixed.

⁴Java™ Platform Standard - [Swing](#)

```

type=eyetracking:point
Long tc           – timecode (milliseconds since the epoch)
String device     – device name
Integer x         – point position x axes (pixels)
Integer y         – point position y axes (pixels)
Boolean fixed (optional) – fixation information

```

Type Pupils Sizes

The Pupils Sizes type contains the information describing the size of the left and the right pupil respectively for each device for a defined timecode. It contains the diameter size of each eye in millimetres.

```

type=eyetracking:pupils
Long tc           – timecode (milliseconds since the epoch)
String device     – device name
Double left       – left pupil diameter size (millimetres)
Double right      – right pupil diameter size (millimetres)

```

Type Time

This simple type is just representing the time by containing only a timecode in milliseconds. This message can be sent at anytime on the bus by the Controller.

```

type=eyetracking:time
Long tc           – timecode (milliseconds since the epoch)

```

Type Device

The Device type is describing the display the user is looking at. The type contains the width and the height in pixels of the screen.

```

type=eyetracking:device
String device     – device name
Integer width     – device width (pixels)
Integer height    – device height (pixels)

```

Type Fixation

This type represents a fixation in the sense of the eye-tracking eye movement analysis for a certain device according to a timecode. The type contains the fixation position on the screen but

also the mean radius, the max radius and the duration of the fixation.

```

type=eyetracking:fixation
Long tc                – timecode (milliseconds since the epoch)
String device          – device name
Integer x              – fixation position x axes (pixels)
Integer y              – fixation position y axes (pixels)
Integer meanradius     – mean fixation radius (pixels)
Integer maxradius      – maximum fixation radius (pixels)
Long duration          – fixation duration (milliseconds)

```

Type Zone

Finally, the zone type, is a data representing a zone of interest to analyse relatively to the timecode for a certain device name. There are three possible zone types, the rectangular, the elliptical or the circular zone. The zone type contains different kinds of messages. There are messages to draw zones with the shapes just mentioned above, but also messages to remove particular zone (identified by the zone name) or to remove all displayed zones. As a result of the second upgrade of the Eye Tracking Monitor, the variable **String widget** has been added in order to identify the widget component type of the zone for further filtering.

```

type=eyetracking:zone
Long tc                – timecode (milliseconds since the epoch)
String device          – device name
String type (ZoneCircle, ZoneEllipse, ZonePoint, ZoneRectangle, ZoneToRemove, ZoneToRemoveAll) – zone type
String name (for all but ZoneToRemoveAll) – zone name
String widget (for all but ZoneToRemoveAll) – zone widget type
Integer x (for ZoneCircle, ZoneEllipse, ZonePoint) – zone position center point (pixels)
Integer y (for ZoneCircle, ZoneEllipse, ZonePoint) – zone position center point (pixels)
Integer x1 (for ZoneRectangle) – zone position upper left point (pixels)
Integer y1 (for ZoneRectangle) – zone position upper left point (pixels)
Integer x2 (for ZoneRectangle) – zone position lower right point (pixels)
Integer y2 (for ZoneRectangle) – zone position lower right point (pixels)
Integer r (for ZoneCircle) – radius (pixels)
Integer a (for ZoneEllipse) – semi major axis (pixels)
Integer b (for ZoneEllipse) – semi minor axis (pixels)

```

Type Fixation in Zone

The fixation in zone type describes the zone that a user fixed during a certain amount of time for certain devices relatively to the timecode.

```

type=eyetracking:fixinzone
Long tc           – timecode (milliseconds since the epoch)
String device     – device name
String name       – zone name
Long duration     – fixation in zone duration (milliseconds)

```

4.6.1 The Cognitive Workload Type

The implementation of the Usybus Cognitive Load Monitor agents leads to the apparition of two new data types. The first data type is the load type that represents the data of the right and left ICA for certain devices relatively to the timecode.

```

type=eyetracking:load
Long tc           – timecode (milliseconds since the epoch)
String device     – device name
Double lICA       – left index of cognitive activity
Double rICA       – right index of cognitive activity

```

4.6.2 The Task type

This new type also appeared for the needs of Usybus Cognitive Load Monitor agents. The task name type is the data to describe a task with an associated task name for a certain devices relatively to the timecode.

```

type=eyetracking:load
Long tc           – timecode (milliseconds since the epoch)
String device     – device name
String taskname   – task name

```

4.6.3 Types usage overview

The Table 4.1 is a summary of the utilisation of the type by agents. The "R" letter signifies that corresponding agents are Receiving the data type. While the letter "E" signifies that associated agents are Emitting this data type on the Usybus.

Aside from the Message Player agent, the table does not contain any other agents that are emitting Zone type. The reason is that the data Zone type is emitted by external java `JFrame`⁵ applications. In order to send data to the Usybus those applications should import the **Usybus Widgets Tracker** library. This library contains among others the class `JFrameConnector`. When instantiated with a `JFrame` object as argument and optionally with the device name, the

⁵ Java™ Platform Standard - [JFrame](#)

	Eye Tracking Monitor	Pupils Sizes Monitor	Eye Tracking Analyser	Controller Agents	Messages Recorder/ Player	Cognitive Load Monitor
Point	R		R	E	R/E	
Pupils Sizes		R		E	R/E	
Time	R			E	R/E	
Device	R	R	R	E	R/E	
Fixation	R		E		R/E	
Fixation in Zone	R		E		R/E	
Zone	R		R		R/E	
Load				E	R/E	R
Task				E	R/E	R and E

Table 4.1: Usage Overview of the Usybus Types

java application will send data zones type from the content pane of that **JFrame** to the Usybus. More specifically, all the widget component to which a name as been assigned. The **Swing** widget components children of the **JFrame** are automatically tracked, thus if the application window is resized, minimised or moved (**Swing** widgets are resized, minimised or moved), new data zone types are sent to the Usybus to update to zone position of the widget.

4.7 AUML Formalisation for Agents Specification

The Usybus model is multi-agent architecture where agent interact with each other. Fundation for Intelligent Physical Agents (FIPA) organisation which promotes the interoperability heterogeneous software agent and agent-based systems will be used to express agents interaction protocols. FIPA is a standard organisation agreed and recognised by IEEE⁶ for agents and multi-agent systems.

More especially, Usybus agents interactions will be specified using the Agent Unified Modeling Language (AUML) formalism modelling language based on Unified Modeling Language (UML) and compatible with the FIPA standard that focus on the behaviour of agents [GV09]. The Figure 4.12 illustrates the sequence diagram applying AUML of communication between agents through the Usybus. Both receiving and sending agent first need to connect to the Usybus. Once it's done, the receiving agent must bind each type of message data it wants to retrieve from the Usybus. After that, this agent enters in a listening loop for any incoming messages from the desired data types. At any time the agent can unbind data types and stop receiving associated messages. In the case of the sending agent, a **UsybusMsg** is created for any message transmitted to the Usybus that will then be transmitted to receiving agent. The agents both ending with a disconnect message to the Usybus when all operations are over.

You would notice that the sequence diagram using AUML in this case looks similar to an ordinary UML sequence diagram. The reason is that Usybus agents are independent of other agents. There is a blind dialogue between agents, that means agents do not react and make a decision based on other agent's actions. Sending agent will send data on the bus without worrying if other agents are binding this type of data or even connected. Conversely, receiving agents can bind or unbind data types whether or not a sending agent is emitting data or is connected.

⁶Institute of Electrical and Electronics Engineers - <https://www.ieee.org>

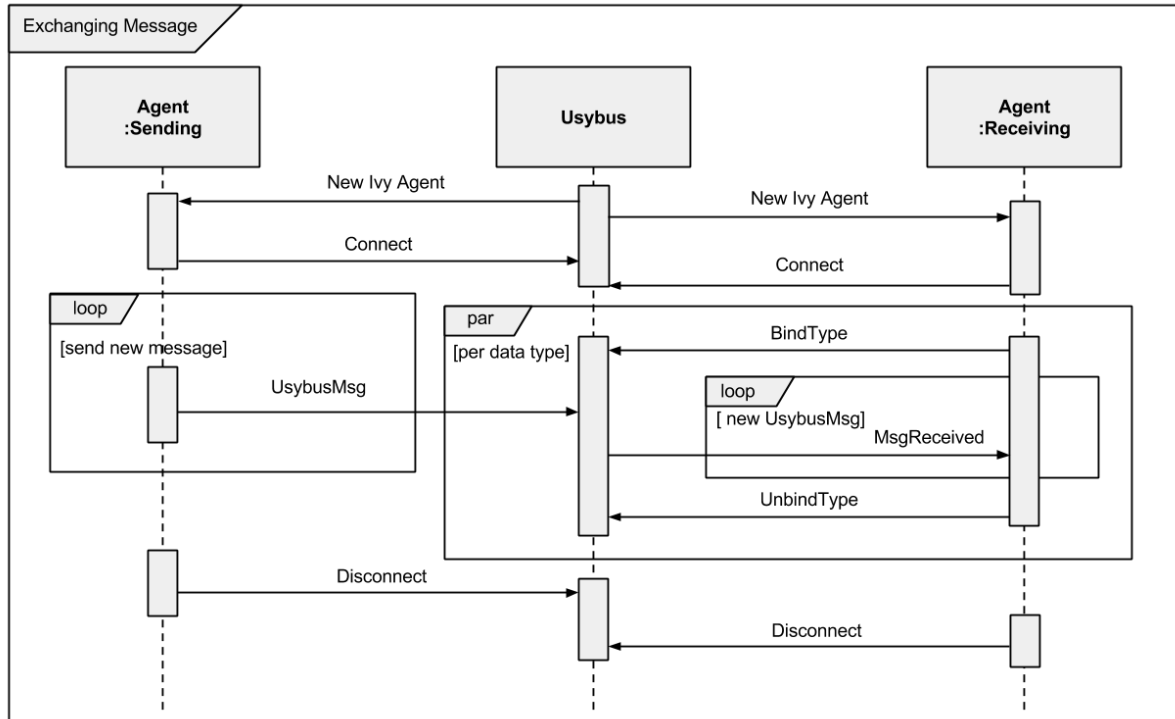


Figure 4.12: Usybus Agents Exchanging Message Interaction Sequence Diagram

4.8 User Interface Adaptation as Usybus Agent

Now having all the necessary tools, our first objective was to test the Usybus architecture. Integrating eye tracking analysis to applications that try to improve the usability of computer interfaces could enlarge the field of research in Human-Computer Interaction. For this experiment we focused on Adaptive User Interface (AUI), advocating the flexibility of UI to adapt to the individual user needs.

The challenge that is facing researchers and developers is to design adaptive computers interface to suit the preferences of the user. More personalized user interaction experiences involve lots of artificial intelligence analysis and machine learning to capture the real needs of the user. Moreover in the context of multimodal systems, describing the user-system interaction which is a sequence of low-level events is essential in order to trigger adaptation of the interface [Bez08]. Those user-system interactions are the actions performed by the user through haptic input device or speech input. In any case, the actions of the user will be interpreted by the system as a form of feedback.

There are two types of feedback as mentioned in [JSK10].

- The feedback of the user can be explicit with ratings scale available to the user. This type of feedback is very accurate concerning the user preference, but not abundant and requires a certain effort from the user.
- The second sort of feedback is implicit and is collected by observing the user behaviour. Typically this feedback will be observable and measurable parameters extracted from the user during the utilisation of the software application. While that feedback in general are abundant, they are less accurate than explicit feedback.

Using the eye-tracking technology, we can utilise brain activity or gaze point as an additional

source of information for implicit feedback.

It has been possible to integrate a AUI software to the Usybus architecture. This software was developed under the context of the Louvain School of Management research institute project ([Mez13], [MLMV15], [MV15], [KMV15]) on Adaptive Interface based on User Feedback.

The Figure 4.13 illustrates the integration of multimodal AUI in a broader sense to the Usybus architecture.

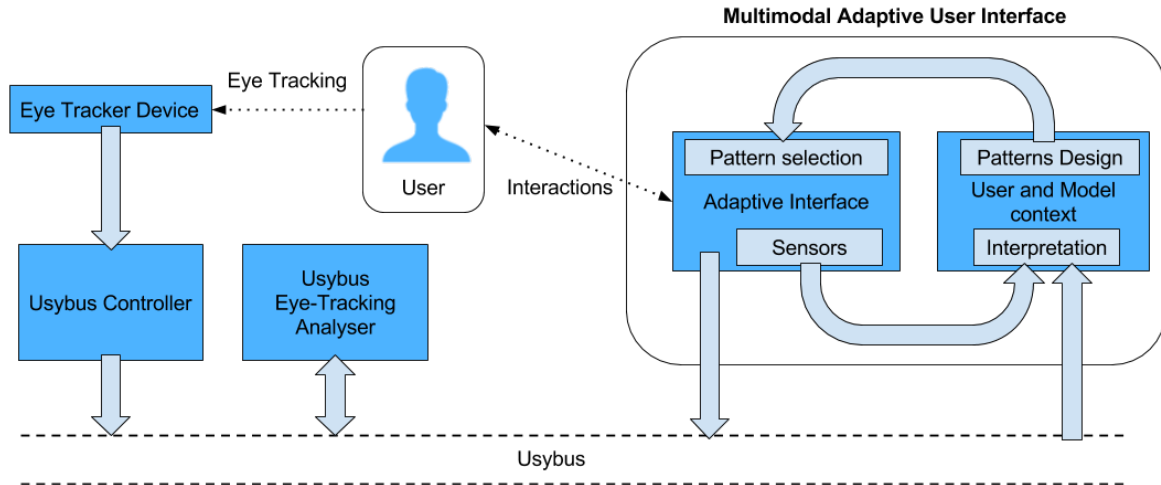


Figure 4.13: Adaptive Interface as Usybus Agent Schema

The schematic shows the Usybus Controller and Eye-Tracking Analyser agents exchanging data on the Usybus as specified in Table 4.1, whereas the Adaptive Interface is sending zones from the UI widget by instantiating the `JFrameConnector` class mentioned in Section 4.6.3. The interactions between the user and the machine are captured through sensors of the multimodal system. Those user-system interactions combined with visual data from the Usybus and are interpreted in the User and Model context. The analysis results in the design of different patterns that are selected to adapt the UI. Among the patterns that are selected, the adaptive interface could for example increase the size of widget elements of the UI that have been missed by the user. With some learning process, the adaptive application could change the location of the widgets according to the needs of the user or decide which widget should be selected under the circumstances.

4.8.1 The PDA-LPA Design Space

More generally, eye tracking in adaptive user interfaces is entirely consistent with the design space for UI adaptation introduced by Bouzit et al. [BCC⁺17].

This design space is explicitly based on the Perception-Decision-Action (PDA) cycle from theory of psychological perception augmented by the Learning-Prediction-Action (LPA) cycle for supporting the adaptation. The PDA-LPA design space illustrated in Figure 4.14, shows the PDA cycle, with perception, decision, and action activities making it possible for the user or the system to make sense of its environment and to interact with it.

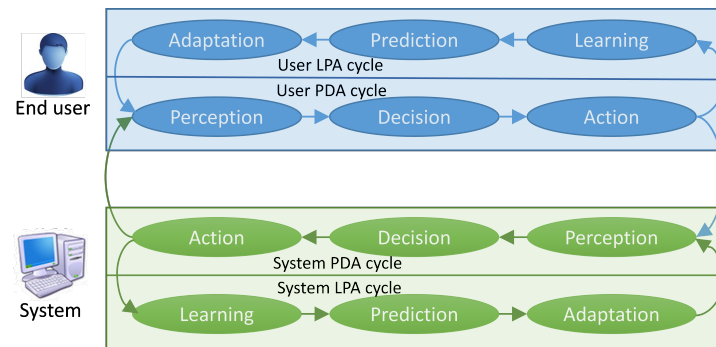


Figure 4.14: The PDA-LPA Design Space

Source: Orange Labs [BCC⁺17]

On one hand, the end user in the perception activity uses the visual information among others, in order to be aware of the system state and actions. On the other hand, the system in the perception activity with the help of eye tracking can perceive any eye movement actions of the user. Those eye movements can be context-aware actions in the case of interactive applications (see Section 3.2.2), but can simply be data that will be used in a learning process to identify the user behaviour.

Chapter 5

UsiTracker for Conception and Evaluation of User Interface

Having now our software environment to perform HCI research in the eye tracking domain, we can focus on a tool to evaluate how eye tracking could improve UI design evaluation.

This chapter will use the UsiTracker tool to perform HCI usability evaluation. UsiTracker (for User Interface Tracker) is a User Interface design software where the user is able to create its own interface and test it with a simple click. The software also monitors a collection of data such as session data and visual data. Hence, the UsiTracker is totally compatible with the Usybus software architecture. The software is therefore ideal to quickly design UI, to gather data, to evaluate it and finally to redesign and modify the UI when necessary.

5.1 Designing the User Interface

During the design step, the user is invited to generate the UI. In order to do that, he has different tools at his disposal. The first one is the Component toolbar.

The Component toolbar is a docked or floating toolbar containing five sections of different buttons allowing the user to select a **Swing**¹ widget (see Figure 5.1). The first section is useful to add text components like labels, text fields, password fields and text areas. The second section allows users to create buttons, checkboxes, radio buttons and toolbars. While the third is for containers like tabbed containers, scroll-bar containers or grids. Next we have structures as trees, tables, combo boxes or lists. Finally, we have special components with separators, progress bars, sliders, spinners, and date fields.

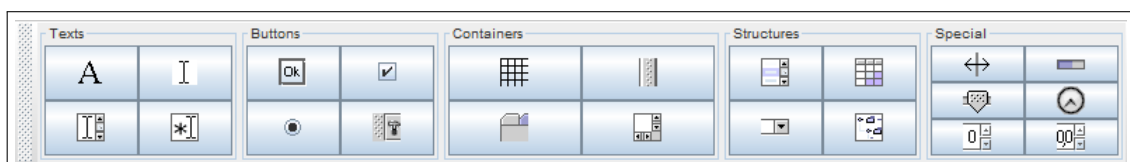


Figure 5.1: UsiTracker Component Toolbar

To choose the location of those components the user has a first grid element at his disposal with four rows and four columns. The cells are represented in blue in Figure 5.2. With a

¹Java™ Platform Standard - [Swing](#)

right click on the grid, the user can modify its composition by adding, removing a row or a column.

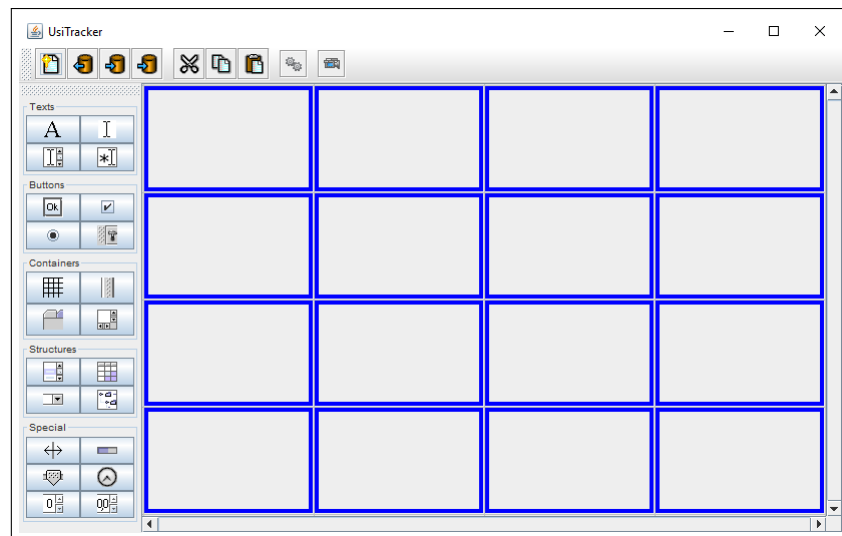


Figure 5.2: UsiTracker Main Window Interface

When adding a widget the user is able to configure it. A component dialogue window opens where the user can define the different parameters for the widget. The user can modify latter the parameters by clicking right on the component located on the grid cell. Each widget has its own parameters. For example, with the button widget illustrated in Figure 5.3, we can edit the name, the text of the button, the tool tip. We can add a mnemonic² (command shortcut), change the position, but also set an icon for the button looking and/or for different events. In this case the user can also set a command action string that will be associated with the button. On the bottom of the dialogue window, there is a general overview of how the widget looks like.

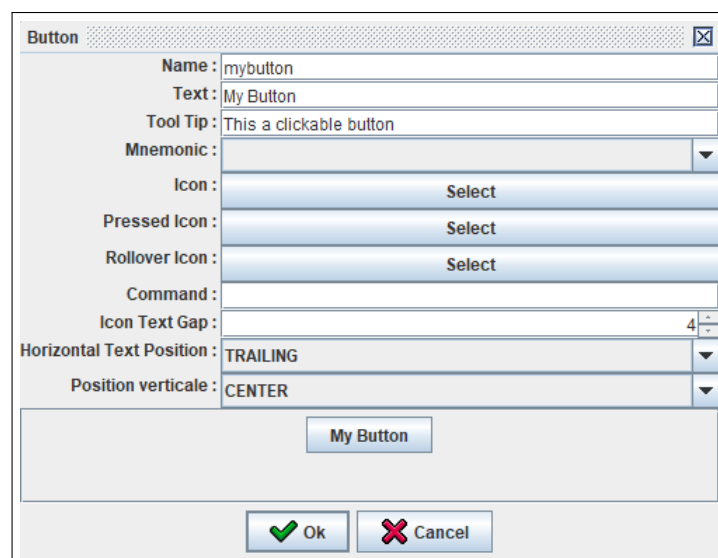


Figure 5.3: UsiTracker Components Dialogue Window

Moreover, the user can use the menu toolbar on the top of the main window of the UsiTracker as illustrated in Figure 5.4 and separated in two sections defined by the framed red borders. In

²The command shortcut is defined by the CTRL+"the mnemonic value" key combination.

this toolbar the user will find the usual Cut, Copy and Paste function (the second section) that can be used when selecting components of the created UI. Additionally, the standard keyboard shortcuts for Cut, Copy and Paste are also enable.



Figure 5.4: UsiTracker Menu ToolBar

In the first section of the menu toolbar in chronological order we can: start editing a new UI, load a template, save the current template and finally save the UI template to Java file in order to export the **Swing** template to any Java project. The last two buttons will be discussed respectively in Section 5.3 and 5.2. The first one is the "Settings" button and the second one is the "Test" button for the evaluation of the User Interface.

5.1.1 XML Representation for User Interface Templates

As outlined above, every created UI template can be saved, the resulting output file is saved as XML format file. This file keeps the memory of the structure of the UI template and the configurations of each widget. Those XML files are formally described and should respect the set of rules expressed by its XML Schema Definition (XSD) in order to be considered conform. We will describe below this XSD following recommendation of the World Wide Web Consortium (W3C)³ using UML diagram class.

In the first place, after the prolog, the root element that must be found on every XML template is the element `<form>` (see Figure 5.5). This first element must contain one child element `<grid>`, this grid is the first grid the user has at his disposal to design the UI.

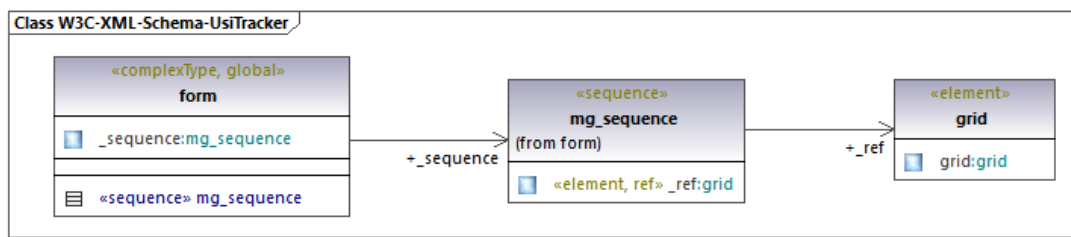


Figure 5.5: Form Element - UML Class Schema

The `<grid>` element (see Figure 5.6) represent the `GridBagLayout`⁴ layout manager that aligns components by maintaining rectangular grid of cells. It contains two required attributes, the `dx` and the `dy` attributes. Those attributes are `nonNegativeInteger` indicating the number of columns (`dx`) and the number of rows (`dy`). The `<grid>` element also contains two implied attributes (optional attributes), `fixedx` and `fixedy`. Those two boolean attributes are indicating respectively if the grid number of columns or the grid number of rows is fixed or not. The `<grid>` element has for children the `<wrapper>` elements that can occur zero, once or several times.

³W3C - <https://www.w3.org>

⁴Java™ Platform Standard - `GridBagLayout`

By default the values of the boolean variables are set to false and thus the grid his not fixed. This will be the case for any other boolean attributes we will encounter during the description of the XML Schema Definition.

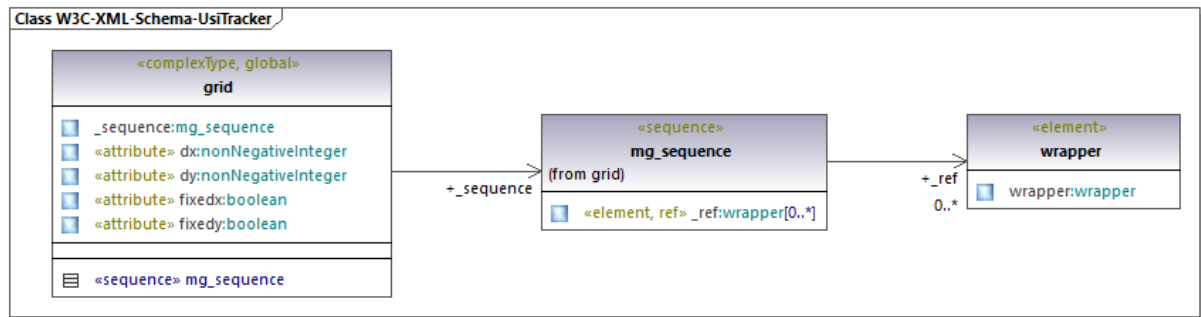


Figure 5.6: Grid Element - UML Class Schema

The `<wrapper>` element (see Figure 5.7) represent any widget type. In reality it represents a component of the `GridBagLayout` root's grid. The `<wrapper>` element has always for children the `<constraint>` element and the `<component>` element. Additionally, it can contain zero or one times both the `<content>` element and the `<border>` element. The `<wrapper>` element has three optional attributes that are the XML tag string, the mask string and lastly the mandatory boolean attribute. Those attributes are optional, because they are not applicable to any type of widget. Except for the tag attribute that is just a XML label for the widgets, but that is not compulsory. Mask and mandatory attribute are useful for texts type widgets. The former specifies the valid characters the text field can contain, while the latter specifies if it is mandatory or not to fill the text field.

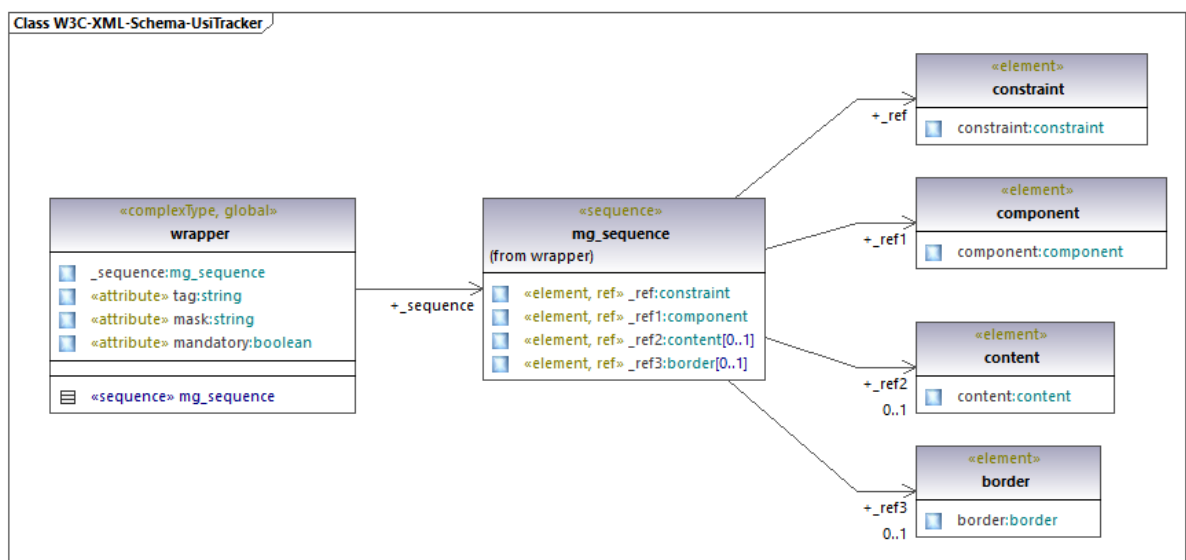


Figure 5.7: Wrapper Element - UML Class Schema

The `<wrapper>` element (or display area) is a component occupying one or more cells. The child element `<constraint>` (see Figure 5.8) that represent a `GridBagConstraints`⁵ will accordingly define the meta data and positioning for this wrapper. The `<constraint>` element only contains optional attributes except for the `nonNegativeInteger` `gridx` and `gridy` attributes that

⁵ Java™ Platform Standard - [GridBagConstraints](#)

represent the coordinate of the cell inside the grid.

To quickly describe the other attributes, we have the nonNegativeInteger **gridwidth** and **gridheight** specifying number of column and row cells the display area will fill. The **anchor** that determine where to place the component within the wrapper. The **anchor** NMTOKEN type is a valid XML name among the following list of absolute values:

CENTER, NORTH, NORTHEAST, EAST, SOUTHEAST, SOUTH, SOUTHWEST, WEST, and NORTHWEST.

The double attributes **weightx** and **weighty** define the horizontal weight and vertical weight of the wrapper that specifies how to distribute extra space. While **fill** indicates how the component's wrapper will be resized to fill its wrapper. This time the **fill** NMTOKEN type is valid for the four values:

NONE, HORIZONTAL, VERTICAL, and BOTH.

Finally, we have the internal and external padding with the four **insets**, the **ipadx** and the **ipady** nonNegativeInteger attributes.

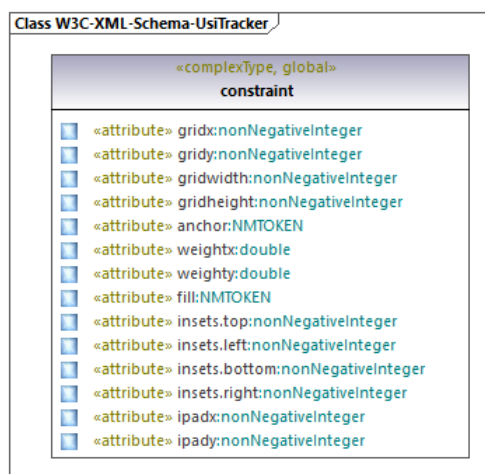


Figure 5.8: Constraint Element - UML Class Schema

The **<component>** element (see Figure 5.9) represents the widget component inside the display area. It has the **class** NMTOKEN attributes that represents a list of valid widgets class types. Those classes are the available widgets to design the UI that we discussed above in Figure 5.1. The element contains sequence of **<param>** elements that can occur zero, once or several times.

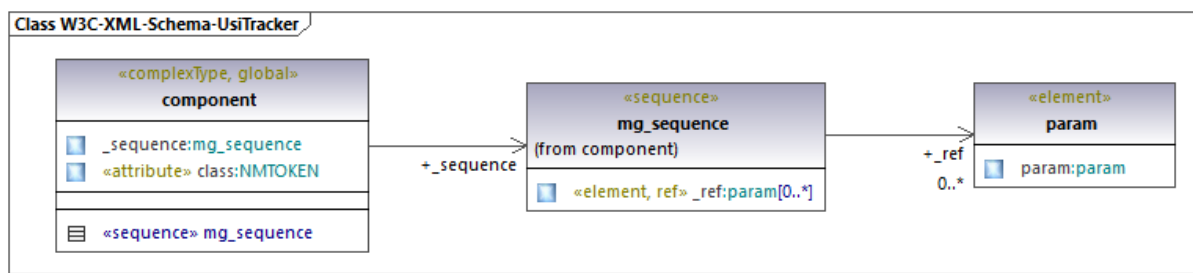


Figure 5.9: Component Element - UML Class Schema

The `<param>` elements (see Figure 5.10) are the parameters for the corresponding component's root. It extends the string class since this XML element can contain text content. They all contain the NMToken attributes `name` and `class`. The former is the name of the widget parameter (e.g. `ToolTip`, `Icon`, `Text`, `Mnemonic`, `Selected`, etc.), while the latter is the type of the parameter value (e.g. `java.lang.Boolean` `java.lang.Double` `java.lang.Integer` `java.lang.String`, etc.). The content of the `<param>` elements is thus the value of those parameters variable.

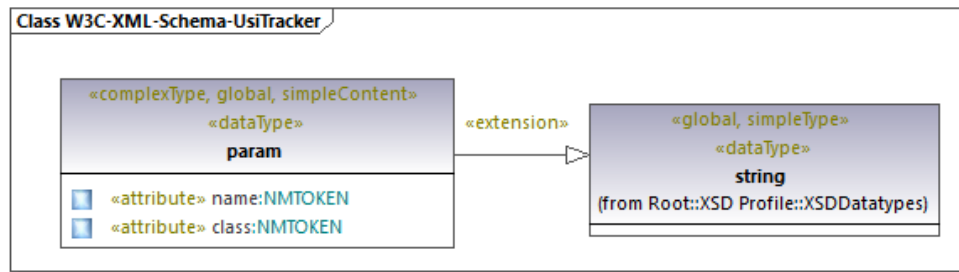


Figure 5.10: Param Element - UML Class Schema

The `<content>` element (see Figure 5.11) is present as a child in the `<wrapper>` element only if the component inside this display area is a container, e.i widgets that can contain other widgets like split pane, grid, scroll pane, toolbar and tabbed pane. `<content>` elements have the option between two types of children, the `<grid>` or the `<tab>` element. Every containers are `<grid>` container except when the widget is a tabbed pane. In that case, the `<content>` children will be one or more `<tab>` element for each tab of the tabbed pane.

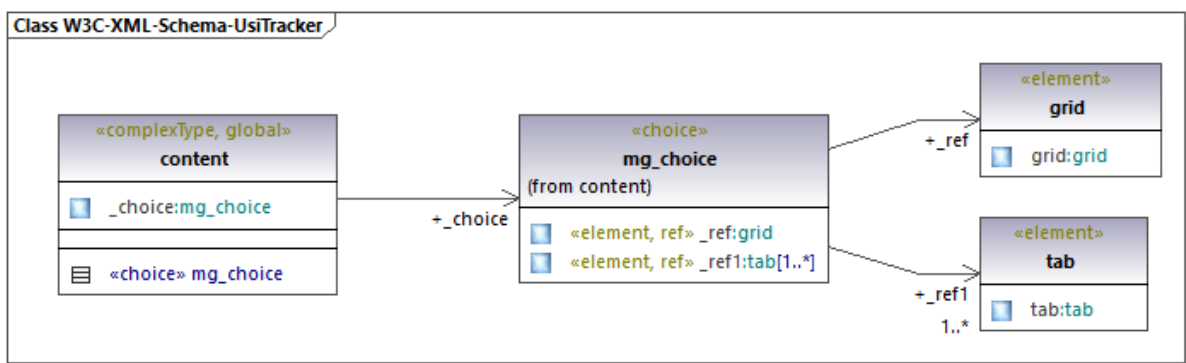


Figure 5.11: Content Element - UML Class Schema

The `<tab>` element (see Figure 5.12) represents one tab of tabbed pane container. Each `<tab>` element contains the `<grid>` element as a child. The nonNegativeInteger `index` attribute is the index of the corresponding tab. That means if the tabbed pane has seven tabs, the range of the index goes from 0 to 6. The tab title is represented by the string attribute `title` while the two last and optional attributes are useful to add a mnemonic or an icon to the corresponding tab. The first NMToken `mnemonic` attributes type is valid if it belongs to the following list of keys:

A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z, 0, 1, 2, 3, 4, 5, 6, 7, 8 and 9.

That mnemonic allows users to easily access to the tab content when pressing the ALT-key and the underlined letter on the keyboard. The second, the string `icon` is simply the path to the icon file that will be displayed in the tab header.

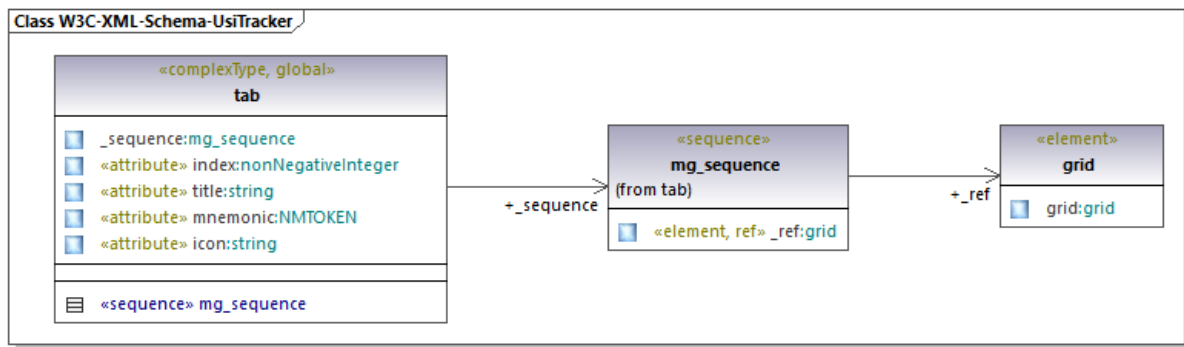


Figure 5.12: Tab Element - UML Class Schema

Finally, The `<border>` element (see Figure 5.13) is present in the `<wrapper>` element if its component has a border. The type NMTOKEN is the list of acceptable border type:

BevelBorder, TitledBorder, EmptyBorder and EtchedBorder.

The `<border>` extends the string class, thus it can contain text content. The content is the title of the border in the case of `TitleBorder`.

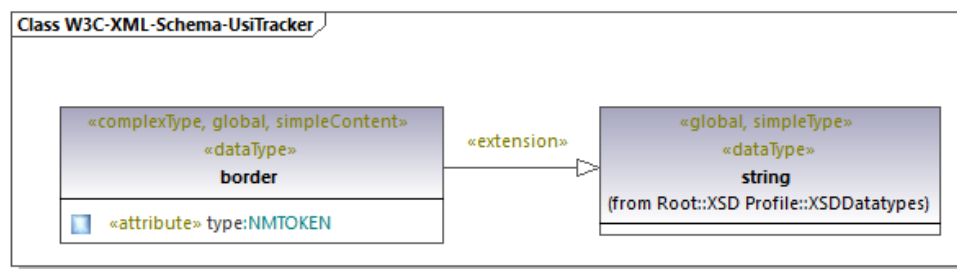


Figure 5.13: Border Element - UML Class Schema

5.2 Evaluating the User Interface

Once a template has been created or loaded, the UsiTracker's user can test the UI by clicking on the last button of the menu tool bar. This button is the camera button previously introduced and illustrated in Figure 5.4. This is the evaluating step, where a new viewing window containing the testing User Interface will open. More specifically, this new window is a Java **Swing JFrame** containing the components of the template with all the configurations and parameters defined by the user.

5.2.1 Usybus Widget Naming Standard

Before the window opens when the user decides to test the UI, a dialogue appears. The dialogue ask whether or not the user wants to use the standard Usybus Widget Tracker naming convention. If the answer of the user is positive, a new dialogue opens where the user can configure the widget he wants to rename. The naming dialogue that is illustrated in Figure 5.14

allows the user to choose which widget will be renamed if the corresponding checkbox is selected.

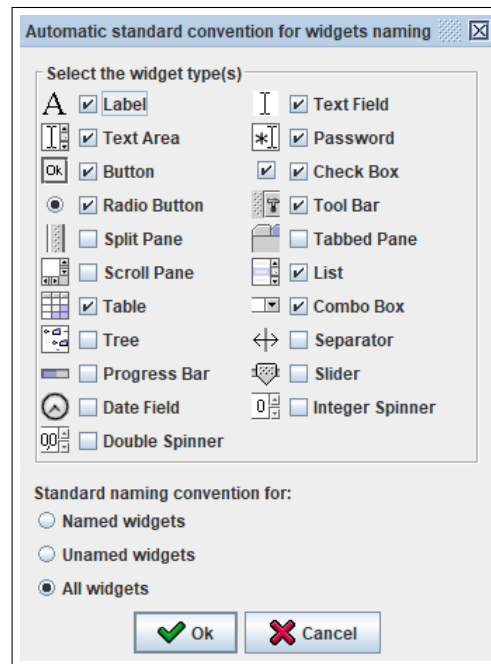


Figure 5.14: Usybus Standard Widget Naming Dialogue

Subsequently, the user has the choice between three options for the naming.

1. **Named widgets:** when this option is selected, the naming convention will be applied only to widgets that already have an instantiated name and that are selected in the widget list above.
2. **Unnamed widgets:** if that option is selected, the naming convention will be applied only to widgets that do not have an instantiated name and that are selected in the widget list above.
3. **All widgets:** in this case the naming convention will be applied to all widgets that are selected in the widget list above.

The naming convention in any case will not change the configuration of the template. Since each test instantiates a new `JFrame` window, it only sets the name for the current test. Moreover, the history of the naming dialogue is conserved in the user preferences. Therefore, the naming dialogue keeps the last configuration even after a new run of the `UsiTracker` application.

This automatic naming convention feature is important because every named widget will be tracked on the `Usybus` as a zone of interest. To stay consistent and to provide uniformity, the naming convention will help to easily identify the type of widgets that are tracked. Widget renaming format is composed of three segments and follow the composition: `<Pre>:<Abb>:<Post>` that stand for the prefix, the abbreviation and the postfix.

- **<Pre>:** The prefix is the widget type abbreviations (e.i. `BTN` for buttons, `TXF` for text field, `PSW` for passwords, `CHX` for checkboxes, etc.). The full list of supported widgets can be found in [Appendix B.1](#) within the Java enum `WidgetType`.
- **<Abb>:** The abbreviation is taken from the label associated with the widget. If there is no associated label, the abbreviation is set to the full name of the type of the widget (e.i.

JButton for buttons, JTextField for text field, JPasswordField for passwords, JCheckBox for checkboxes, etc.).

- **<Post>**: The postfix is the identifier. This is a sequentially numbered integer relative to the type of the widget. Meaning that the range goes from one to the total number of that widget type.

The naming convention is implemented in the **Usybus Widget Tracker** library we already discussed (Section 4.6.3). This library encompasses the class **UsybusWidgetNaming** that contains static method **nameComponents** to apply the naming convention to the component structure.

UsybusWidgetNaming class

```

1  /**
2   * Naming the components with the Usybus Widget Tracker naming convention
3   * @param component, the root component
4   * @param widgetsList, the list of widget is the components type to rename
5   * @param replace, naming convention for named components
6   * @param autofill, naming convention for unnamed components
7   */
8  public static void nameComponents(Component component, ArrayList<WidgetType>
9      widgetsList, boolean replace, boolean autofill);
10
11  // Return the prefix of the component
12  private static String getPrefix(Component component);
13
14  // Return the abbreviation of the component
15  private static String getAbbreviation(Component component);
16
17  //Return the identifier of the component
18  private static String getIdentifier(Component component);
19
20  //Return true if the component name respect the naming convention
21  private static boolean isNamingFormat(Component component);
22
23  //Returns the list of all Components of a component structure,
24  private static ArrayList<Component> getAllComponents(Component component);

```

5.3 Recording the UsiTracker Session

The running session of the UsiTracker records a series of data produced by the user during the design and the evaluation of the UI. This data set also contains visual measurements collected when the user is testing the UI. The settings of the recording can be accessed by clicking on the last button of the menu toolbar, the gearing button.

The Figure 5.14 illustrates the settings dialogue window for the recordings. First the user can choose whether or not to activate the saving of the session when exiting the application. However, he can save the data during a running session by clicking the "Save" button. In both cases the user can choose the folder location on his computer. Saving the session while running the UsiTracker will re-initialise the current session to a new session as if we launched a new instance of the UsiTracker. Nevertheless, the current active template remains unchanged.

There are two types of data that can be recorded by the UsiTracker. The measurements are saved under CSV file format for easier readability and data handling. The first type is the data from the UsiTracker session itself. Those raw data are the number of widgets of the template, the number of widgets for each type, the total session time, the number of operations done by

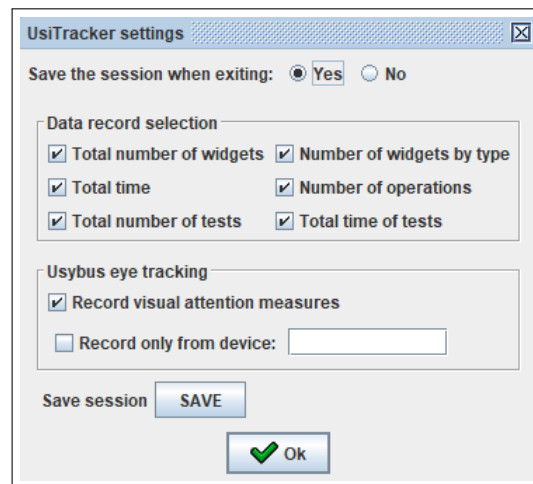


Figure 5.15: UsiTracker Settings Dialogue

the user, the number of tests and finally the total time of those tests.

The second type of data are the visual measurements that are only recorded during the testing session for evaluation of the UI. The visual attention measures are calculated from the data on the Usybus. The user can choose whether or not to record those data and to set a filtering devices name.

Among those visual measurements we will find descriptive data for the ICA for the totality of the testing session like the mean (**MICA**), the median (**MedICA**), the max (**MaxICA**) and the min (**MinICA**) value, first and third quartile (**Q1ICA** and **Q3ICA**) and the standard deviation (**SDICA**).

However, those data are also classified by tasks, that means those descriptive data are calculated for each defined task during the testing session. Those tasks are monitored by the Usybus Cognitive Workload we discuss in Section 4.4. Indeed, we saw in Section 3.5.1 that usability evaluation requires participants to execute given tasks during the testing session. Therefore, having statistical visual data for each task is important to identify usability issues concerning those tasks.

Moreover, in order to characterise participants' visual attention patterns, we will also record quantitative measures introduced by Radu-Daniel VATAVU from the University Stefan cel Mare of Suceav in Romania [VM15].

5.3.1 The Visual Attention for Multi-screen

VATAVU developed the Visual Attention Toolkit for TV (VATic-TV) toolkit that can compute measures from data delivered by standard eye-tracking equipment. This software toolkit proposes new set of measures for understanding visual attention of emerging multi-screen TV applications [VM15].

We can point out there is some correlation between multi-screen TV and multi-widget display from UI. The experimental set-up of Radu-Daniel VATAVU only considers coplanar screen. Therefore, both stimulus are similar in the sense that visual content is in the same vertical plane, but spatially separated. Moreover, in both cases, screen TV and widget display (e.i. the zones of interests) are relatively close to each other. That means our head does not need to move to

visualise all the displays, only a few degrees of eye rotation are sufficient. As TV screens do, we can assume that widget-displays visually attract and catch more the attention of participants than the rest of the UI that remains more neutral.

However, we are facing some constraints that we must consider by using the visual measure toolkit. First, we need to keep in mind that there is a significant difference between the number of displays. While in multi-screen TV we will almost never exceed four screens, in widget situation we will end-up with far more displays. Additionally TV screens remain fixed, so we must assume that our widget will remain fixed. Indeed, if the stimulus content is dynamic, it becomes another configuration and the calculations are no longer aligned. Therefore, the UsiTracker does not allow the participant to move or resize its testing application if the record of visual attention measures is activated.

VATic-TV paper introduces six new objective measures that have been integrated and adapted to the Usybus for evaluation:

- **Discovery time (DT):** the amount of time the participant takes to discover all the different widget displays. Meaning that each display has been fixated visually at least once.
- **Discovery sequence (DS):** the sequence of displays the viewer passes over during the discovery time. This measure can therefore be a permutation of all widget name. However, the sequence can also include the same widget viewed more than once. This is the case when that display attracted the participant's visual attention more than one times during the discovery time.
- **Screen watching time (SWT):** the percentage of visual attention allocated to each widget during the monitored time. The more attractive the display widget content, the more the participant will devote time to the display and higher the SWT percentage will be.
- **Transition count (TC):** the number of visual transitions between widget display to another during the monitored time interval. Larger TC values implies more distributed attention over all the widgets.
- **Eye gaze travel distance (EGTD):** the total distance travelled by the participant's eye gaze during the monitored time interval. This measure is expressed with the units reported from the oculometer, in our case it is in pixels.
- **Switch time (ST):** the percentage of time the participant spent between widget display transition during the monitored time interval. In other words, this is the percentage of the monitored time where the viewer didn't fix any widget display. The larger the ST value, the more the widgets are far from each other. Thus, larger ST might indicate flaws in the design of the UI.

5.4 Usitracker for User Interface Design Evaluation Cycle

In Human-Computer Interaction, most systems development efforts and a large amount of time and resources are dedicated to the designing of User Interface [MR92]. Most projects are using User Interface development tools, however, at each stage of the development, evaluation of the UI need to be performed. Evaluation is necessary to rapidly identify usability problems in the design or to adapt the software to the user needs. The Figure 5.16 illustrates the User

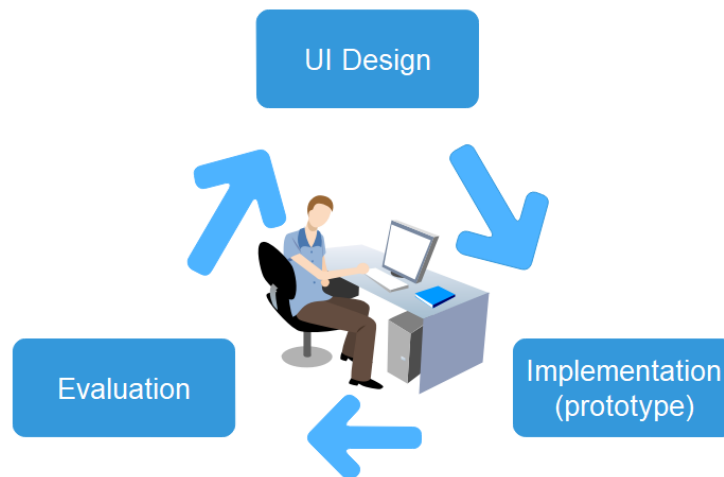


Figure 5.16: User Interface Design Evaluation Cycle

Interface design evaluation cycle for each stage of the development lifecycle.

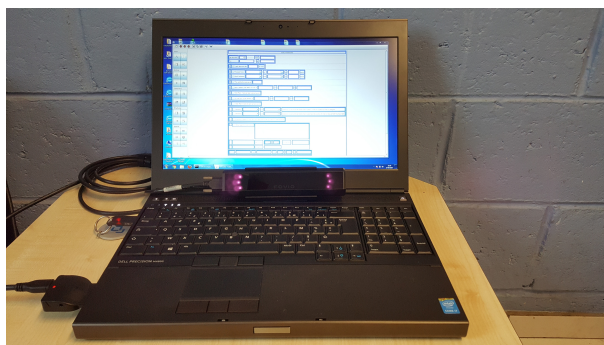
The usability evaluation is an important part of the overall user interface design process [IH01]. However several factors limit its use, usability evaluation might be some time expensive and time-consuming compare the benefit it provides, especially in the early stage of the development lifecycle. However, it's often too late for significant changes when recommendation comes only at a late stage of the development [JMWU91]. Moreover, using eye tracking in usability evaluation add an additional workload for developers. At each iteration of the UI, the we need to install and model the experiment, calibrate the system, conduct the experiment, collect the results, interpret the results, make recommendations for improvement.

The UsiTracker attempts to reduce those costs by integrating the evaluation directly during the development phase. Using UsiTracker could enhance the efficiency of the usability evaluation protocol schematics, we discuss in Section 3.5.1. Indeed, iteration for each participant is unified in a single application process and quantitative data are automatically computed and ready for the analysis. UsiTracker put in perspective the emergence of a new type of user interface development tools that will provide effective and quantitative measurement for usability evaluation.

5.5 UsiTracker Test Case with SmallApp Template

A small test case has been carried out in order to show how the UsiTracker tool can perform in a convenient way for eye-tracking UI usability evaluation even at an early stage of development. Five voluntary (all male) participated in the experiment with a range of ages from 20 to 24. The participants self-reported the ease of usage of the computer and software application program.

The set-up utilised for the experiment was consisting only of the FOVIO oculometer, a computer with the UsiTracker, the EyeWorks software (for the calibration) and the Usybus EyeWorks Controller (we described in Section 4.3). In this test case we only launched the minimal requirements to conduct the experiment correctly. However, we could have used any other agents that would be useful for the experiment such as the Usybus Eye Tracking Monitor to visualise and save pictures of the session or the Usybus Cognitive Workload to visualise ICA and monitor tasks.



The picture shows the TrackWarrant⁶ template designed by the UsiTracker ready for evaluation and the non-intrusive FOVIO oculometer that is located just below the computer screen.

Figure 5.17: Example of UsiTracker Eye Tracking Evaluation Environment

The template UI designed for the test case is named SmallApp and is illustrated in Figure 5.18. This UI is consisting of a label "Insert text here:" followed by an empty text field and the two buttons "Ok" and "Cancel" at the bottom.

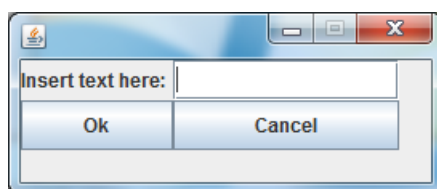


Figure 5.18: UsiTracker SmallApp User Interface

For each user session, there was only a single task assigned to the participant because of the simplicity of the UI. It was only requested to write any text the participant wants in the text field and to confirm or cancel the operation with one of the two buttons. Using the UsiTracker each participant could evaluate the UI by clicking on the "Test" button from the menu toolbar and ended the session by closing the tested UI window.

The participants didn't need to make the calibration, which is only possible with the UsiTracker tool. Instead, the calibration has been performed only once with the EyeWorks software. There is a loss of quality by not calibrating for each of the participants, however that loss is acceptable regarding to the gain of time it procures. Nevertheless, the calibration can still be done between each evaluation session when more precision is needed.

widgets	time	operations	tests	timetests	txf	btn	lab
4	00:09:19	10	5	00:02:33	1	2	1

Table 5.1: UsiTracker SmallApp Instance Data Session

Indeed, as the Table 5.1 shows, the time for the whole UsiTracker session only last 9 minutes and 19 seconds, e.i. to design the UI (which takes ten operations) and to evaluate it. While the

⁶Track warrants are railway network permissions issued by dispatchers to train crews instead of using signalisation.

time evaluating the UI was 2 minutes and 33 seconds (± 30 seconds for each participant).

The UsiTracker computed the VATic-TV measure for each session of the participants (represented by the entry `test_1`, `test_2`, `test_3`, `test_4` and `test_5`) as illustrated in Figure 5.19. The Discovery Sequence measures show that most of the time participants looked first to the text field widget which makes sense since the widget occupies an important role in the execution of the task. In the second place users were next focused on the "Cancel" button, one of the reasons could be that because the widget attracts more visual attention.

	A	B	C	D	E	F	G
1	File	DT	DS	SWT	TC	EGTD	ST
2	test_1	10.19	TXF:myTextField:1 ~ BTN:cancel:2 ~ LAB:mylabel:1 ~ BTN:ok:1	7.890223 ~ 20.926243 ~ 1.3722128 ~ 1.7152658	20	10799.617	68.096054
3	test_2	10.422	TXF:myTextField:1 ~ BTN:cancel:2 ~ LAB:mylabel:1 ~ BTN:ok:1	5.832148 ~ 31.578945 ~ 0.85348505 ~ 5.263158	13	4814.9595	56.472267
4	test_3	21.932	TXF:myTextField:1 ~ BTN:cancel:2 ~ LAB:mylabel:1 ~ BTN:ok:1	3.0763528 ~ 13.454411 ~ 0.11119348 ~ 2.2238696	24	33218.926	81.13418
5	test_4	2.449	TXF:myTextField:1 ~ BTN:cancel:2 ~ LAB:mylabel:1 ~ BTN:ok:1	0.6961364 ~ 6.7873306 ~ 1.4966933 ~ 3.0630004	38	38772.684	87.95685
6	test_5	17.97	BTN:cancel:2 ~ TXF:myTextField:1 ~ LAB:mylabel:1 ~ BTN:ok:1	3.526971 ~ 7.261411 ~ 0.48409408 ~ 5.1175656	17	18448.71	83.60995

Figure 5.19: UsiTracker SmallApp VATic-TV Visual Measurements

Moreover, the Screen Watching Time for all participants indicates that the text field widget (which is the second widget on the UI in the direction of reading) has the largest percentage of visual attention allocated. It reaches more than 31% for the second participant. Followed by the label widget for the first three participants, certainly due to the reading function the label provides. On the contrary the fourth participant appears to have not even taken the time to read the label. In general the Switch Time is relatively high for all participants, this can be explained because the SmallApp application does not occupy lot of the screen space. Users are therefore tempted to look to other visual stimuli all over the computer screen.

H	I	J	K	L	M	N
MICA	SDICA	MedICA	Q1ICA	Q3ICA	MaxICA	MinICA
0.15249999	0.06520212	0.14250001	0.09999999	0.21	0.275	0.055
0.12416666	0.0503548	0.1225	0.08	0.145	0.215	0.035
0.16322918	0.06506535	0.15	0.12000001	0.2	0.33499998	0
0.17365384	0.066289	0.16499999	0.115	0.215	0.32	0.055
0.16134614	0.07184091	0.14250001	0.115	0.21000001	0.32	0.035

Figure 5.20: UsiTracker SmallApp ICA Visual Measurements

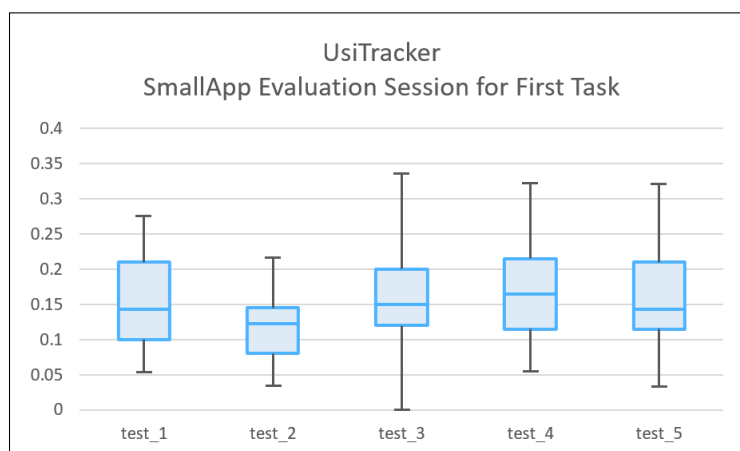


Figure 5.21: UsiTracker SmallApp ICA Visual Measurements Boxplot

The Figure 5.20 illustrates the UsiTracker ICA measures for each participant during the

evaluation session. For better visualisation of those ICA data, boxplots have been created (see Figure 5.21). The tendency shows that the ICA of the participants remains relatively low during the testing session. This means unsurprisingly that relatively low cognitive activity was needed to carry out the task. With a lower median and smaller dispersion of the ICA the second participant seems to have a greater ease in the achievement of the task. In summary, no major flows have been detected, except that widgets could be a little more spatially separated. Moreover, the UI should be bigger and the widgets should take more space as indicated by the relatively low Screen Watch Time measure in overall.

Chapter 6

Conclusion and Perspectives

6.1 Summary of the Problematic

The aim of this dissertation was to explore the different solutions and enhancements that could provide eye tracking for HCI especially in the context of the railway training simulation. Eye tracking like any new emerging technology still needs to prove itself before being adopted and establish itself as an indispensable IT tool. Among the perspectives that offers eye tracking we have highlighted three application domains that are all compatible with railway training simulation.

- The first one is the utilisation of eye tracking as diagnostic approach concerning human factors in railway environment. Reducing the human error and increasing the safety is a key priority for railway companies. Simulator equipped with such technology could help in the analysis of operators' behaviour reacting to simulated events, and in identifying design flaws in the conception of rolling stock operating environment.
- The second approach is the utilisation of eye tracking as an interactive tool during the railway training simulation. Interactivity in the learning process could enhance the performance of the trainee to successfully complete his exercises. As an example, smart tooltip could be displayed when the trainee missed important elements like railway signalisation.
- The last one is the utilisation of eye tracking for usability evaluation. More specifically, the design and the evaluation of User Interface taking into account visual measurement collected from the oculometer. This application domain is therefore particularly appropriate to railway simulation as all training simulator software needs User Interface.

We decided to develop in more details the third domain since it could potentially have a larger scale of use and because UI evaluation is less context-based. By performing an eye-tracking usability evaluation on the R³ software from Transrub Simulation we could defined the experimental protocol for conducting eye tracking evaluation experiment.

However, it turns out that traditional software distributed by companies are not scalable as desired. Additionally, there is a need for new tools to make experimental research in this domain. It is in this context that we introduced firstly the Usybus software architecture as a software environment independent of the oculometer (allowing to link the HCI of the simulator). Then secondly, the UsiTracker as a tool to design and evaluate any HCI independently of the oculometer. By performing some operations using eye-tracking technology and applying certain assumptions and manipulations, it is possible to shorten the duration of a design-evaluation cycle.

6.2 Pros and Cons of the UsiTracker Tool

The UsiTracker software is easy to set-up and simple to use. Even users with low computer skills could handle the design of User Interface. Nevertheless, for evaluation session it is required that the eye tracker device is set-up beforehand by a more experienced person.

The software provides more detailed results because the UsiTracker relies on the Usybus software environment. The evaluation output is more refined since each widget of the HCI is tracked via the Usybus.

The UsiTracker/Usybus environment allows real-time visualisation and post-analysis of visual measurement, the solution is thus flexible and can be adapted to many experiments in the eye tracking HCI research field.

However, limitations of the UsiTracker software start when the design of the User Interface requires complex widgets and more sophisticated elements. The developer is therefore limited by what the UsiTracker can provide as features for developing the HCI.

Finally, UsiTracker is still at a development stage, some functionality and quantitative measurements for UI evaluation are still missing. As we said there is still a lack of eye-tracking quantitative data for usability evaluation and no automatic interpretation of measurements exist.

6.3 Improvements and Perspective

6.3.1 The Usybus Architecture

Concerning the Usybus environment, as long as there is a need for the processing of visual data, there will always be new agents that will be created and added to the architecture. More sophisticated agents might see the light of day, we have already developed an agent that is capable of calculating the detection of fixations. It is most likely that we will need to develop a new agent for processing ICA for the computation of a new visual quantitative measure to refine the overall system.

Furthermore, to remain coherent and consistent with the structure of the Usybus, we could have developed an Usybus agent for tasks scheduling apart from the Usybus Cognitive Workload agent. This agent would have the same specification, that is sending in real time on the Uysbus the indication of the start of a new task or scheduling different tasks, but would be independent.

The lightweight and scalable multi-agent architecture might inspire new development projects in the domain of data processing. The Usybus architecture is an ideal tool for any third party application that needs to exchange visual information.

6.3.2 The UsiTracker Softwares

Potential further improvement of the UsiTracker would be to propose more relevant quantitative data that would be recorded during testing session and provide a semi or full diagnostic output about the tested UI.

Additionally, more complex research need to be performed to validate the performance and the sustainability of the UsiTracker. One feasible approach would be to apply the "mutation testing" method, which consists of producing a bad example of UI design expecting that the tool will be able to detect it. The protocol would be based on an example of a predetermined graphical HCI, but sufficiently complex to be subjected to a "mutation testing". After introducing intentionally design flaws, we could ask two distinct groups of participants to improve as much as possible the User Interface. One group with the UsiTracker at his disposition, the other one without. To prove the effectiveness of the Usitracker, the remaining design flaws of the group using the tool should be by far lower than that of the group not using the tool.

6.3.3 The Graph Analyser

In the context of a project development in Transurb Simulation as a future employee within the company, I will have the opportunity to work on the ongoing Graph Analyser project that will become a new functionality of the R³ software. The Graph Analyser will allow the instructor to analyse visually all the huge amount of data generated and recorded during an exercise. The graph will show Continuous Data (evolving continuously over time and distance) and Event Data (occur only at a specific time and/or locations). The Figure 6.1 illustrates the sketch of the Graph Analyser specifications.

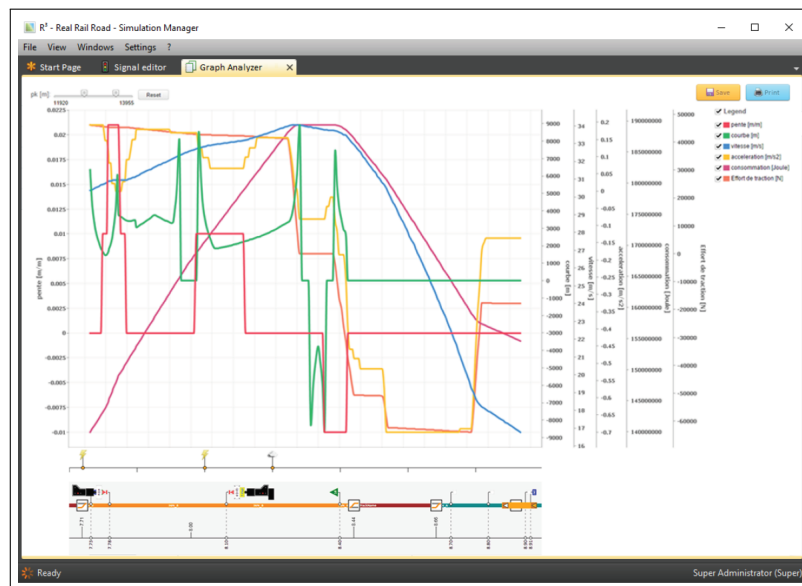


Figure 6.1: The Graph Analyser Sketch Specification

Therefore, we could add to the Continuous Data specifications the value of the ICA evolving over time and distance. The measures that can be observed while running the simulation or in the report of the trainee, would indicate the level of mental effort that the trainee has to provide for a given distance/time of the simulation. This measure could be compared to other cognitive workload references for the same section of the network in order to identify if the trainee has some difficulties for that section.

The Graph Analyser combined with the measure of the Index of Cognitive Activity, would be a first step towards the integration of eye tracking to railway simulator. As far as we know that would be the first real and useful application of eye-tracking technology for users of railway simulator.

References

Bibliography

- [AWS06] Richard Atterer, Monika Wnuk, and Albrecht Schmidt. Knowing the user's every move: User activity tracking for website usability evaluation and implicit interaction. In *Proceedings of the 15th International Conference on World Wide Web, WWW '06*, pages 203–212, New York, NY, USA, 2006. ACM.
- [BBC⁺02] Marcellin Buisson, Alexandre Bustico, Stéphane Chatty, Francois-Régis Colin, Yannick Jestin, Sébastien Maury, Christophe Mertz, and Philippe Truillet. Ivy : un bus logiciel au service du développement de prototypes de systèmes interactifs. In *IHM 2002, 14ème Conférence Francophone sur l'Interaction Homme-Machine*, pages pp 223–226, Poitiers, France, November 2002.
- [BBI07] Brian P. Bailey, Chris W. Busbey, and Shamsi T. Iqbal. Taprav: An interactive analysis tool for exploring workload aligned to models of task execution. *Interact. Comput.*, 19(3):314–329, May 2007.
- [BCC⁺17] Sarah Bouzit, Gaelle Calvary, Joelle Coutaz, Denis Chêne, Eric Petit, and Jean Vanderdonckt. The pda-lpa design space for user interface adaptation. In *Research Challenges in Information Science (RCIS), 2017 11th International Conference on*, pages 353–364. IEEE, 2017.
- [Bez08] Matthias Bezold. Describing user actions in adaptive interfaces with probabilistic acceptors. In Melanie Hartmann, Daniel Krause, and Andreas Nauerz, editors, *16th Workshop on Adaptivity and User Modeling in Interactive Systems*, 2008.
- [BFH10] Matthias Baldauf, Peter Fröhlich, and Siegfried Hutter. Kibitzer: A wearable system for eye-gaze-based mobile urban exploration. In *Proceedings of the Augmented Human International Conference*, 2010.
- [BI02] Richard Bates and Howell Istance. Zooming interfaces!: enhancing the performance of eye controlled pointing devices. In *Proceedings of the fifth international ACM conference on Assistive technologies*, pages 119–126. ACM, 2002.
- [CBCA08] Craig A Chin, Armando Barreto, J Gualberto Cremades, and Malek Adjouadi. Integrated electromyogram and eye-gaze tracking cursor control system for computer users with motor disabilities. 2008.
- [CBD02] Laura Cowen, Linden J. Ball, and Judy Delin. *An Eye Movement Analysis of Webpage Usability*. Springer-Verlag Ltd., 2002.
- [Cle07] Jesper Clement. Visual influence on in-store buying decisions: an eye-track experiment on the visual influence of packaging design. *Journal of Marketing Management*, 23(9-10):917–928, 2007.

- [CLH11] Weitao Che, Hui Lin, and Mingyuan Hu. Reality-virtuality fusional campus environment: An online 3d platform based on opensimulator. *Geo-spatial Information Science*, 14(2):144–149, 2011.
- [CNPB12] Peter M Corcoran, Florin Nanu, Stefan Petrescu, and Petronel Bigioi. Real-time eye gaze tracking for gaming design and consumer electronics systems. *IEEE Transactions on Consumer Electronics*, 58(2), 2012.
- [CWC14] Su Cai, Xu Wang, and Feng-Kuang Chiang. A case study of augmented reality simulation system application in a chemistry course. *Computers in Human Behavior*, 37:31 – 40, 2014.
- [DLBR97] Scott L. Delp, Peter Loan, Cagatay Basdogan, and Joseph M. Rosen. Surgical simulation: An emerging technology for training in emergency medicine. *Presence: Teleoperators and Virtual Environments*, 6(2):147–159, 1997.
- [DS07] Heiko Drewes and Albrecht Schmidt. Interacting with the computer using gaze gestures. *Human-Computer Interaction–INTERACT 2007*, pages 475–488, 2007.
- [Duc07] Andrew T Duchowski. Eye tracking methodology. *Theory and practice*, 328, 2007.
- [GH10] Joseph H. Goldberg and Jonathan I. Helfman. Visual scanpath representation. In *Proceedings of the 2010 Symposium on Eye-Tracking Research & Applications*, ETRA ’10, pages 203–210, New York, NY, USA, 2010. ACM.
- [GK99] Joseph H Goldberg and Xerxes P Kotval. Computer interface evaluation using eye movements: methods and constructs. *International Journal of Industrial Ergonomics*, 24(6):631–645, 1999.
- [GNM15] Sayani Ghosh, Tanaya Nandy, and Nilotpal Manna. Real time eye detection and tracking method for driver assistance system. In *Advancements of Medical Electronics*, pages 13–25. Springer, 2015.
- [GTCC08] Mirta Galesic, Roger Tourangeau, Mick P. Couper, and Frederick G. Conrad. Eye-tracking datanew insights on response order effects and other cognitive shortcuts in survey responding. *Public Opinion Quarterly*, 72(5):892–913, 2008.
- [GV09] Gilleanes Thorwald Araujo Guedes and Rosa Maria Vicari. *Applying AUML and UML 2 in the Multi-agent Systems Project*, pages 106–115. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.
- [HAMM14] John Paulin Hansen, Alexandre Alapetite, I Scott MacKenzie, and Emilie Møllenbach. The use of gaze to control drones. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, pages 27–34. ACM, 2014.
- [HLR14] Emily Higgins, Mallorie Leinenger, and Keith Rayner. Eye movements when viewing advertisements. *Frontiers in Psychology*, 5:210, 2014.
- [HWS15] Graeme Harrison, Marie-Claude Weyber, and Julian Smith. Assessing the global transport infrastructure market: Outlook to 2025. Technical report, 2015.
- [IH01] Melody Yvette Ivory and Marti Hearst. *An empirical foundation for automated web interface evaluation*, volume 1. University of California, Berkeley, 2001.
- [Jam13] Francis Jambon. Analyse oculométrique on line de prémices de fixations. *Activités humaines, Technologies et Bien être*, pages 309–312, 2013.

- [JL12] Francis Jambon and Vanda Luengo. Analyse oculométrique "on-line" avec zones d'intérêt dynamiques: application aux environnements d'apprentissage sur simulateur. In *Proceedings of the 2012 Conference on Ergonomie Et Interaction Homme-machine, Ergo'IHM '12*, pages 1:1–1:8, New York, NY, USA, 2012. ACM.
- [JMWU91] Robin Jeffries, James R Miller, Cathleen Wharton, and Kathy Uyeda. User interface evaluation in the real world: a comparison of four techniques. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 119–124. ACM, 1991.
- [JSK10] Gawesh Jawaheer, Martin Szomszor, and Patty Kostkova. Comparison of implicit and explicit feedback from an online music recommendation service. In *Proceedings of the 1st International Workshop on Information Heterogeneity and Fusion in Recommender Systems, HetRec '10*, pages 47–51, New York, NY, USA, 2010. ACM.
- [KHR04] Mika Koivisto, Jukka Hyönä, and Antti Revonsuo. The effects of eye movements, spatial attention, and stimulus features on inattention blindness. *Vision Research*, 44(27):3211–3221, 2004.
- [KL13] Chulhan Kim and Tae-Eog Lee. Modelling and simulation of automated manufacturing systems for evaluation of complex schedules. *International Journal of Production Research*, 51(12):3734–3747, 2013.
- [KMV15] I. Khaddam, N. Mezhoudi, and J. Vanderdonckt. Adapt-first: A mde transformation approach for supporting user interface adaptation. In *2015 2nd World Symposium on Web Applications and Networking (WSWAN)*, pages 1–9, March 2015.
- [KW13] K. Kurzhals and D. Weiskopf. Space-time visual analytics of eye-tracking data for dynamic stimuli. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2129–2138, Dec 2013.
- [KWKJ15] Bronislaw Kapitaniak, Marta Walczak, Marcin Kosobudzki, and Zbigniew Józwiak. Application of eye-tracking in drivers testing: a review of research. 2015.
- [LÇK10] Xia Li, Arzu Çöltekin, and Menno-Jan Kraak. *Visual Exploration of Eye Movement Data Using the Space-Time-Cube*, pages 295–309. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.
- [Mar02] Sandra P Marshall. The index of cognitive activity: Measuring cognitive workload. In *Human factors and power plants, 2002. proceedings of the 2002 IEEE 7th conference on*, pages 7–7. IEEE, 2002.
- [Mar08] S.P. Marshall. Mental alertness level determination, March 18 2008. US Patent 7,344,251.
- [Mez13] Nesrine Mezhoudi. User interface adaptation based on user feedback and machine learning. In *Proceedings of the Companion Publication of the 2013 International Conference on Intelligent User Interfaces Companion, IUI '13 Companion*, pages 25–28, New York, NY, USA, 2013. ACM.
- [MF01] Tirin Moore and Mazyar Fallah. Control of eye movements and spatial attention. 98:1273–6, 02 2001.
- [MLMV15] Nesrine Mezhoudi, Jorge Luis, Pérez Medina, and Jean V. An agile adaptation framework for context-aware user interfaces, 2015.

- [MR92] Brad A Myers and Mary Beth Rosson. Survey on user interface programming. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 195–202. ACM, 1992.
- [Mur06] Atsuo Murata. Eye-gaze input versus mouse: Cursor control as a function of age. *International Journal of Human-Computer Interaction*, 21(1):1–14, 2006.
- [MV15] N. Mezhoudi and J. Vanderdonckt. A user’s feedback ontology for context-aware interaction. In *2015 2nd World Symposium on Web Applications and Networking (WSWAN)*, pages 1–7, March 2015.
- [PB05] Alex Poole and Linden J. Ball. Eye tracking in human-computer interaction and usability research: Current status and future. In *Prospects”, Chapter in C. Ghaoui (Ed.): Encyclopedia of Human-Computer Interaction. Pennsylvania: Idea Group, Inc*, 2005.
- [PHG⁺04] Bing Pan, Helene A. Hembrooke, Geri K. Gay, Laura A. Granka, Matthew K. Feusner, and Jill K. Newman. The determinants of web page viewing behavior: An eye-tracking study. In *Proceedings of the 2004 Symposium on Eye Tracking Research & Applications, ETRA ’04*, pages 147–154, New York, NY, USA, 2004. ACM.
- [PKSH10] Oskar Palinko, Andrew L Kun, Alexander Shyrokov, and Peter Heeman. Estimating cognitive load using remote eye tracking in a driving simulator. In *Proceedings of the 2010 symposium on eye-tracking research & applications*, pages 141–144. ACM, 2010.
- [PN09] Kara Pernice and Jakob Nielsen. How to conduct eyetracking studies. *Nielsen Norman Group, Fremont, CA*, 2009.
- [RA14] Marc Resnick and William Albert. The impact of advertising location and user task on the emergence of banner ad blindness: An eye-tracking study. *International Journal of Human-Computer Interaction*, 30(3):206–219, 2014.
- [Raj09] S. Rajasekaran. *Bioinformatics and Computational Biology: First International Conference, BICoB 2009, New Orleans, LA, USA, April 8-10, 2009, Proceedings*. Lecture Notes in Bioinformatics. Springer, 2009.
- [RC11] Jennifer C Romano and Jennifer M Chen. A usability and eye-tracking evaluation of four versions of the online national survey of college graduates (nscg): Iteration 2. *Survey Methodology*, 1, 2011.
- [SG00] Dario D. Salvucci and Joseph H. Goldberg. Identifying fixations and saccades in eye-tracking protocols. In *Proceedings of the 2000 Symposium on Eye Tracking Research & Applications, ETRA ’00*, pages 71–78, New York, NY, USA, 2000. ACM.
- [SJ00] Linda E Sibert and Robert JK Jacob. Evaluation of eye gaze interaction. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 281–288. ACM, 2000.
- [ŠM07] O Špakov and Darius Miniotas. Visualization of eye gaze data using heat maps. *Elektronika ir elektrotechnika*, pages 55–58, 2007.
- [SM08] M Schmitz and C Maag. Benchmarking report on computer-based railway training in europe. 2train project report. *Center for Traffic Sciences at the University of Wuerzburg, Wuerzburg*, 2008.

- [VDPSR98] H. Van Dyke Parunak, Robert Savit, and Rick L. Riolo. *Agent-Based Modeling vs. Equation-Based Modeling: A Case Study and Users' Guide*, pages 10–25. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998.
- [VM15] Radu-Daniel Vatavu and Matei Mancas. Evaluating visual attention for multi-screen television: measures, toolkit, and experimental findings. *Personal and Ubiquitous Computing*, 19(5-6):781–801, 2015.
- [VTBG12] MéLodie Vidal, Jayson Turner, Andreas Bulling, and Hans Gellersen. Wearable eye tracking for mental health monitoring. *Comput. Commun.*, 35(11):1306–1311, June 2012.
- [Win15] Eric Winsberg. Computer simulations in science. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, summer 2015 edition, 2015.
- [YWL13] Chung-san Yu, Eric Min-yang Wang, Wen-Chin Li, and Graham Braithwaite. Application of eye-tracking techniques in human factor research in marine operations. challenges and methodology. 108:116–120, 2013.
- [YWL14] Chung-san Yu, Eric Min-yang Wang, Wen-Chin Li, and Graham Braithwaite. Pilots' visual scan patterns and situation awareness in flight operations. 85:708–714, July 2014.

Webography

- [Eur17b] Eurostat. *Railway transport - passengers transported, quarterly data (detailed reporting only) (1 000 pass., million pkm)*. http://appsso.eurostat.ec.europa.eu/nui/show.do?dataset=rail_pa_quartal&lang=en, July 2017. [Online; accessed 22-July-2017].
- [Pen06] Matteo Penzo. *Label Placement in Forms*. <http://www.uxmatters.com/mt/archives/2006/07/label-placement-in-forms.php>, July 2006. [Online; accessed 3-June-2017].
- [Wik17a] Wikipedia. *"Eye tracking — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/wiki/Eye_tracking, 2017. [Online; accessed 07-July-2017].
- [Wik17b] Wikipedia. *Smooth pursuit — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/wiki/Smooth_pursuit, 2017. [Online; accessed 08-July-2017].

Iconography

- [Acu17] Acuity. *Tobii X60 and X120*. http://www.acuity-ets.com/images/products_x-series.jpg, 2017. [Online; accessed 8-June-2017].
- [Eur17a] Eurostat. *Quarterly evolution of rail passenger transport performance at EU28 level (million passenger-kilometres)*. [http://ec.europa.eu/eurostat/statistics-explained/index.php/File:Quarterly_evolution_of_rail_passenger_transport_performance_at_EU28_level_\(million_passenger-kilometres\).png](http://ec.europa.eu/eurostat/statistics-explained/index.php/File:Quarterly_evolution_of_rail_passenger_transport_performance_at_EU28_level_(million_passenger-kilometres).png), January 2017. [Online; accessed 22-July-2017].

- [HIT17] HIT. *Tobii 1750*. http://hit.psy.unipd.it/sites/dipartimenti.it/files/tobii_1750_1.jpg, 2017. [Online; accessed 8-June-2017].
- [iMo17a] iMotions. *How do users respond to novel interfaces and new tech?* <https://cdn.imotions.com/wp-content/uploads/2014/11/Screen-Shot-2015-08-03-at-11.40.23-930x523.png>, 2017. [Online; accessed 11-June-2017].
- [iMo17b] iMotions. *Tobii T60 and T120*. <https://cdn.imotions.com/wp-content/uploads/2012/08/Tobii-T60.jpg>, 2017. [Online; accessed 8-June-2017].
- [Neu17a] Neurolize. *Eye Tribe Tracker Device*. https://www.neurolize.com/App_Themes/Arki/images/devices/dc9f9d9f-a36d-493b-9cfe-c7219c89be14/eyetribe3.png, 2017. [Online; accessed 8-June-2017].
- [Neu17b] Neurolize. *Eye Tribe Tracker Packaging*. https://www.neurolize.com/App_Themes/Arki/images/devices/dc9f9d9f-a36d-493b-9cfe-c7219c89be14/eyetribe2.jpg, 2017. [Online; accessed 8-June-2017].
- [oF17] Neurolab University of Ferrara. *Tobii X50*. <http://www.unife.it/progetto/neurolab/image/tobiix50.jpg>, 2017. [Online; accessed 8-June-2017].
- [Sim17a] Transurb Simulation. *Compact Simulator*. http://simulation.transurb.com/sites/default/files/4._alstom_ertms_compact_simulator.jpg, 2017. [Online; accessed 03-June-2017].
- [Sim17b] Transurb Simulation. *Full-Cab Simulator*. http://simulation.transurb.com/sites/default/files/2._nsb_full-cab_simulator.jpg, 2017. [Online; accessed 03-June-2017].
- [Sim17c] Transurb Simulation. *Nano Simulator*. http://simulation.transurb.com/sites/default/files/4._ertms_nano_simulator.jpg, 2017. [Online; accessed 03-June-2017].
- [Sim17d] Transurb Simulation. *The Norwegian train crew simulator*. http://simulation.transurb.com/sites/default/files/7._nsb_train_crew_simulator.jpg, 2017. [Online; accessed 11-June-2017].
- [Sim17e] Transurb Simulation. *The virtual reality for the safety of railway infrastructure workers*. http://simulation.transurb.com/sites/default/files/uploads/simulation/infrabel_visu_01.jpg, 2017. [Online; accessed 13-July-2017].

Appendices

Appendix A

Hardware specification

A.1 The Eye Tribe device

Eye Tribe Tracker is a very affordable oculometer. The prize of the device is 99\$. As other eye tracking device it is highly mobile and light, therefore the eye tracker requires very short preparation time. It is connected and charged through the same USB cable. Therefore, this device fits better for average consumer and not experiment software user.



Figure A.1: The Eye Tribe device

Source: Neurolize [[Neu17b](#)], [[Neu17a](#)]

Product Specification

- Sampling rate: 30 Hz to 75 Hz
- Accuracy: $0.5^{\circ} - 1^{\circ}$
- Latency: < 16 ms
- Screen sizes: Up to 27"

A.2 The Tobii 50 Series devices

The Tobii 50 Series is composed of the Tobii 1750, X50, T60, T120, X60 and X120. The section below will briefly describe the oculometer devices.

A.2.1 The Tobii 1750

The Tobii 1750 is remote eye tracker allowing to monitor remotely eye behavior and pupil diameter. The device is totally none invasive and moreover it is highly tolerant to head movements. The Tobii 1750 was one of the first oculometer from Tobii.



Figure A.2: The Tobii 1750 device

Source: HIT [[HIT17](#)]

Product Specification

- Sampling rate: 50 Hz
- Accuracy: 0.5°
- Freedom of Head Movement: 30x16x20cm at 60cm from tracker
- Drift: $< 1^\circ$
- Latency: 35 ms
- Screen sizes: 17"
- Max gaze angles: 40°

A.2.2 The Tobii X50

This device without requirement for extensive eye tracking experience is available to anyone. The Tobii X50 is one of the first oculometer not integrated into a monitor. Therefore, can be attached to any monitor or used to perform eye tracking relative to a physical scene, large displays, a TV or projection screen. However, it is slightly more obtrusive. Additionally, each time the device is moved, there is a need for hardware calibration.



Figure A.3: The Tobii X50 device

Source: Neurolab [[oF17](#)]

Product Specification

- Sampling rate: 50 Hz
- Accuracy: 0.5° - 0.7°
- Freedom of Head Movement: 30x16x20cm at 60cm from tracker
- Drift: $< 1^{\circ}$
- Latency: 35 ms
- Max gaze angles: 35°

A.2.3 The Tobii T60 and T120

The Tobii T60 eye tracker and T120 eye tracker are integrated into a 17" TFT¹ monitor. They are non-intrusive and ideal for all forms of eye tracking studies with stimuli that can be presented on a screen. Thus the device is intended only for use in office and home environments.



Figure A.4: The Tobii T60 device

Source: iMotion [[iMo17b](#)]

Product Specification

- Sampling rate: 60 Hz (T60) and 120 Hz (T120)
- Accuracy: 0.5°
- Freedom of Head Movement: 44x22x30cm
- Drift: $< 0.3^{\circ}$
- Latency: 33 ms
- Max gaze angles: 35°
- Screen sizes: 17"

¹Thin Film Transistor

A.2.4 The Tobii X60 and X120

The Tobii X60 and X120 have the same characteristics as the Tobii X50 except that these two oculometers are more efficient in their specifications / performances.



Figure A.5: The Tobii X60 device

Source: Actuity [[Acu17](#)]

Product Specification

- Sampling rate: 60 Hz (X60) and 120 Hz (TX120)
- Accuracy: 0.5°
- Freedom of Head Movement: 44x22x30cm at 70cm from tracker
- Drift: $< 0.3^{\circ}$
- Latency: 30 ms - 35 ms
- Max gaze angles: 35°

Appendix B

The Usybus Widget Tracking Library

B.1 Supported Widget Type

WidgetType enum class

```
1  public enum WidgetType {
2      JLabel ,
3      JTextField ,
4      JTextArea ,
5      JPasswordField ,
6      JButton ,
7      JCheckBox ,
8      JRadioButton ,
9      JToolBar ,
10     JSplitPane ,
11     JTabbedPane ,
12     JScrollPane ,
13     JList ,
14     JTable ,
15     JComboBox ,
16     JTree ,
17     JSeparator ,
18     JProgressBar ,
19     JSlider ,
20     JDateField ,
21     JIntegerSpinner ,
22     JDoubleSpinner ,
23     Unknown;
24
25     public static WidgetType getComponentType(Component component) {
26         WidgetType type = null;
27         try{
28             type = WidgetType.valueOf(component.getClass().getSimpleName());
29         }catch(IllegalArgumentException e){
30             type = WidgetType.Unknown;
31         }
32         return type;
33     }
34
35     @Override
36     public String toString(){
37         String str = null;
38         switch(this){
39             case JLabel:         str = "LAB";
```

```

40         break;
41     case JTextField:      str = "TXF";
42         break;
43     case JTextArea:      str = "TXA";
44         break;
45     case JPasswordField: str = "PSW";
46         break;
47     case JButton:        str = "BTN";
48         break;
49     case JCheckBox:      str = "CHX";
50         break;
51     case JRadioButton:   str = "RDB";
52         break;
53     case JToolBar:       str = "TLB";
54         break;
55     case JSplitPane:     str = "SPP";
56         break;
57     case JTabbedPane:    str = "TBP";
58         break;
59     case JScrollPane:     str = "SCP";
60         break;
61     case JList:          str = "LST";
62         break;
63     case JTable:         str = "TBL";
64         break;
65     case JComboBox:      str = "CBB";
66         break;
67     case JTree:          str = "TRE";
68         break;
69     case JSeparator:     str = "SPT";
70         break;
71     case JProgressBar:   str = "PRB";
72         break;
73     case JSlider:        str = "SLD";
74         break;
75     case JDateField:     str = "DTF";
76         break;
77     case JIntegerSpinner: str = "ISP";
78         break;
79     case JDoubleSpinner: str = "DSP";
80         break;
81     case Unknown:        str = "Unknown";
82         break;
83     default :
84         break;
85 }
86 return str;
87 }
88 }

```