FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Simulation and Management of Environmental Disturbances in Flight Simulator X

João Fernando de Sousa Almeida

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Daniel Castro Silva

September 22, 2017

Simulation and Management of Environmental Disturbances in Flight Simulator X

João Fernando de Sousa Almeida

Mestrado Integrado em Engenharia Informática e Computação

September 22, 2017

Abstract

The frequency of natural and man-made disasters which affect the environment has been rising in recent years. In order to comprehend, examine, and even combat these occurrences, various vehicles are utilized on missions: aircraft, ships and automobiles represent the majority. Such vehicles may be manned, remote-controlled, or even fully autonomous; the latter being increasingly common.

As to provide a way of simulating cooperative multi-vehicle missions, a platform was developed at the Artificial Intelligence and Computer Science Laboratory (Faculty of Engineering, University of Porto) using Microsoft's Flight Simulator X (FSX) as its core and simulation engine. Despite being centered on a flight simulator, it supports vehicles beyond aircraft.

The Platform includes a control panel, which allows for the specification of each mission; an agent to manage traffic; a vehicle control agent; performance analysis; and the basis for a disturbances management tool. The Platform enables the specification and deployment of a wide variety of missions using multiple vehicle types on various simulations. While still technically in prototype stage, the Platform is currently at a fairly advanced level.

Is it possible to create an environmental disturbances manager, and effectively integrate it within the existing platform? Is it feasible to introduce a generic interface for external disaster simulators into the system? The former research questions represent the focus of the research and development work at hand. A positive answer would be an excellent sign that the Platform was effectively improved, and that it became more suitable for a real world application. A neutral or even negative answer may be undesired, however it may still provide a good insight on what is achievable on a similar architecture, and provide the researcher with the knowledge to apply on improving current or future solutions to similar problems.

The Disturbances Manager is responsible for the simulation of environmental anomalies - which may require the intervention of vehicle teams, or have an impact on unrelated vehicular missions; it is launched through the Platform's Control Panel, communicates with FSX and with the Vehicle Control Agents - as to relay the corresponding sensor readings (e.g. an automobile being driven in the vicinity of a burning building will get higher temperature and carbon dioxide readings).

The Control Panel allows for the specification of disturbances using a graphical interface, as well as saving and loading of the resulting XML files. After the mentioned steps, the Disturbances Manager is launched to handle the Disturbances at hand - handling their position, space and time evolutions, generated sensor readings, and more.

Given that the Disturbances Manager was fully integrated into a distributed simulation platform, similar architectures were studied and taken into consideration on how to perform said integration.

A fire may be spreading on a mountain range, which needs to be located; a hurricane may occur in the set course for an upcoming flight. The Disturbances Manager aims to facilitate the simulation of such events, among others.

The main objective for this dissertation was the development the disturbances management component of the mentioned platform, with simulates objective of simulating a wide array of disturbances, such as fires, pollution, storms, volcanic eruptions and chemical spills. The simulation of such disturbances allows the posterior automatic adapting of vehicular missions according to the placement, evolution, and sensor readings related to the disturbances at hand (e.g. to avoid a low-visibility area in a commercial flight or to locate the center of a forest fire, in the case of aerial firefighting).

The Disturbances Manager was implemented within The Platform, achieving the core objective of the dissertation. A battery of tests was performed in order to validate its functionalities and understand its limitations. It was verified that the sensor readings are sent to the vehicles once they reach the affected area; the readings are higher at the center of the disturbances; the time evolutions of the emitted readings occur as planned; the spacial growth of the disturbances' areas also performs as expected.

Keywords: simulation, environmental, disturbances, flight, multi-vehicle, missions

Resumo

A frequência de desastres que afectam o ambiente, tanto naturais como originados pelo Homem, tem vindo a aumentar. De modo a compreender, examinar, e até combater estas ocorrências, vários veículos são utilizados em missões: aeronaves, navios e automóveis representando a maioria. Tais veículos podem conter um condutor humano, podem ser controlados remotamente, ou até ser completamente autónomos, com esta alternativa a tornar-se cada vez mais comum.

De modo a providenciar uma forma de simular missões cooperativas multi-veículo, uma plataforma foi desenvolvida utilizando o *Flight Simulator X* da Microsoft (FSX) como o correspondente núcleo e motor de simulação. Apesar de ser centrada num simulador de vôo, a plataforma suporta veículos para além de aeronaves.

A Plataforma inclui um painel de controlo, que permite a especificação de cada missão; um agente para gerir o tráfego; um agente de controlo de veículos; análise de desempenho; e a base para uma plataforma de gestão de distúrbios. A Plataforma permite a especificação e execução de uma abrangente variedade de missões utilizando múltiplos tipos de veículos em várias simulações. Apesar de ainda tecnicamente em fase de protótipo, A Plataforma encontra-se numa fase bastante avançada.

Será possível criar um gestor de distúrbios ambientais, e integrá-lo eficazmente na plataforma existente? É exequível introduzir uma interface genérica para simulações externas de desastres no sistema? As questões apresentadas representam o foco da tarefa de investigação e desenvolvimento em questão. Uma resposta positiva para estas questões seria ser um excelente sinal de que a Plataforma foi melhorada eficazmente, e que essa se tornou mais adequada para aplicações no mundo real. Uma resposta neutra ou até negativa pode ser indesejada, no entanto providenciaria também uma boa perspectiva sobre o que pode ser alcançado em arquiteturas semelhantes, e fornecer ao investigador conhecimento a ser aplicado no sentido a melhorar soluções atuais ou futuras para problemas semelhantes.

O Gestor de Distúrbios é responsável pela simulação de anomalias ambientais - que podem requerer a intervenção de equipas veículares, ou ter impacto em missões veículares não relacionadas; este é lançado através do Painel de Controlo da Plataforma, comunicando com o FSX e com os Agentes de Controlo de Veículos - de modo a transmitir aos mesmos as leituras dos sensores correspondentes (e.g. um automóvel a ser conduzido na vizinhança de um edifício em chamas obterá leituras de temperatura e de dióxido de carbono acima da norma).

O Painel de Controlo permite a especificação de distúrbios através de uma interface gráfica, tal como guardar e carregar os ficheiros XML resultantes. Após os passos mencionados, o Gestor de Distúrbios é lançado para processar os mesmos - gerindo a respetiva posição, evoluções temporal e espacial, leituras de sensores geradas, e mais.

Sendo que o Gestor de Distúrbios foi integrado numa plataforma de simulação distribuída, arquiteturas semelhantes foram tidas em consideração no que toca a como realizar a integração mencionada.

Um incêndio pode estar a propagar-se numa serra, sendo necessário localizá-lo; um furacão pode ocorrer no trajeto planeado para um vôo prestes a descolar. O objectivo do Gestor de Distúrbios é facilitar a simulações de eventos desse tipo, entre outros.

O objetivo principal desta dissertação foi o desenvolvimento do componente de gestão de distúrbios da plataforma mencionada, com o propósito de simular uma vasta gama de distúrbios, tais como incêndios, poluição, tempestades, erupções vulcânicas e derrames químicos. A simulação de tais distúrbios permite a posterior adaptação automática de missões veiculares de acordo com a localização, evolução e leituras de sensores relacionadas com os distúrbios em questão (e.g. para evitar uma área de baixa visibilidade num vôo comercial ou de modo a localizar o centro de um fogo florestal, no caso de combate aéreo a incêndios).

O Gestor de Distúrbios foi implementado n'A Plataforma, atingindo o objectivo principal da dissertação. Um conjunto de testes foi realizado de modo a validar as respectivas funcionalidades e entender as suas limitações. Foi verificado que as leituras de sensores são enviadas para os veículos assim que estes atingem a área afectada; as leituras são mais elevadas no centro dos distúrbios; as evoluções temporais das leituras emitidas ocorrem como planeado; o crescimento espacial das áreas dos distúrbios também ocorre como esperado.

Palavras-chave: simulação, ambiental, distúrbios, vôo, multi-veículo, missões

Acknowledgements

I would like to start this section by stating my gratitude to my supervisor, Daniel Silva. He was not only the proponent for this dissertation and the creator of The Platform, but also an invaluable and ever-present source of knowledge, advice and meaningful discussions. I am also thankful toward Álvaro Câmara, who is a part of the project and provided many helpful comments, and Vasco Gonçalves, my friend and colleague who helped me review the document.

I would also like to thank all my friends, colleagues and teachers who worked with me and helped me throughout my academic path.

Finally, and most importantly, I would like to give thanks to my dear family and beloved girlfriend for withstanding my frequent absences and, above all, for the unconditional support throughout this journey.

Dedicated to my late father, Fernando Almeida, who passed on too soon.

João Almeida

"And once the storm is over, you won't remember how you made it through,
how you managed to survive.
You won't even be sure, whether the storm is really over.
But one thing is certain.
When you come out of the storm, you won't be the same person who walked in.
That's what this storm's all about."

Haruki Murakami

Contents

List of Figures

LIST OF FIGURES

List of Tables

Abbreviations

API	Application Programming Interface
ATC	Air Traffic Controller
CSV	Comma Separated Values
CPM	Counts Per Minute
CPU	Central Processing Unit
DDL	Disturbances Description Language
DM	Disturbances Manager
ED	Environmental Disturbance
HDD	Hard Disk Drive
FIPA	Foundation for Intelligent Physical Agents
GUI	Graphical User Interface
FSX	Flight Simulator X
HLA	High Level Architecture
IP	Internet Protocol
JSON	JavaScript Object Notation
OS	Operating System
RAM	Random Access Memory
RPM	Rotations Per Minute
SATA	Serial AT Attachment
SDK	Software Development Kit
TDL	Teams Description Language
UI	User Interface
XML	eXtensible Markup Language
XSD	XML Schema Definition

Chapter 1

Introduction

In the present chapter, an overview of the thesis at hand is provided. At first, the context in which the project is inserted is presented, as well as the motivation behind it; then, the main goals and research questions are described; followed by the problems being approached; next, the chosen methodology to do so; and finally an outline of this document's structure.

1.1 Context and Motivation

Considering the increase in the frequency of climatic disasters [CRED, 2015], it is becoming more relevant to develop methods of studying them. In the real world, multiple types of vehicles are utilized on missions related to environmental disasters: from cars to aircraft. As to provide a means of simulating such missions, a platform was developed [Silva, 2011] centered on Microsoft's Flight Simulator X. The Platform is the basis for the work at hand.

Multiple vehicle missions can be simulated through The Platform, such as searching for a specific terrain type, drop missions (e.g. dropping water over an area supposedly on fire or dropping a food crate over an area with a rough terrain) and tracking and escorting an airplane.

One of the original purposes for The Platform is the automation of specific tests and tasks. As an example, at The Platform's initial state, in order to simulate the dropping of water on burning terrain, the user must decide on which area he considers "on fire" (since the platform didn't simulate fires). After such decision, the user should check whether the helicopter travels to the correct location (not identified on the map), and if the drop occurs on the right spot. The described interaction requires user intervention and supervision on many levels. If there were a way to simulate environmental disturbances and programatically deploy them at set times, that would allow for the posterior full automation of such missions - thereby achieving one of The Platform's objectives. Later, a layer of automated testing could be introduced.

The Platform was developed at the Faculty of Engineering of the University of Porto, at the Artificial Intelligence and Computer Science Laboratory. It has been the focus of one Ph.D.

thesis[Silva, 2011] and four completed Masters' theses [Camara, 2013][Sousa, 2010][Santos, 2010][Silva, 2008]; and is currently the basis for one ongoing Ph.D. thesis and three Master's theses. A few articles have also been written around The Platform [Sousa et al., 2010][Câmara et al., 2014][Rodrigues et al., 2015].

The Disturbances Manager aims to add a disturbance simulation component to The Platform, allowing the vehicles therein to become aware of ongoing disturbances near their location. When completed, the DM should be a significant contribution toward not only The Platform's feature set, but also toward its potential regarding future functionalities.

1.2 Main Goals & Research Questions

The primary objective for this dissertation is to implement the component of the mentioned platform responsible for the simulation of multiple types of disturbances (environmental and more) - e.g. forest fires, floods, oil spills or lost people. Attaining the goal of a functional Disturbances Manager (DM) enables the posterior adapting of vehicular missions to specific disturbances. As an example: if an aircraft can get carbon dioxide sensor readings regarding an ongoing forest fire, it may adjust its course as to minimize such readings, or even survey the area as to locate the center of the fire - therefore allowing the testing of multiple search algorithms to find their time efficiency at locating (i.e. putting out) fires. While forest fires were used as an example, the same may be said for analogous disturbances.

The secondary objective is the implementation of a generic interface to be used between the DM and external disaster simulators. If a user wishes to simulate, say, an earthquake using a 3rd-party earthquake simulator, he should be able to use his desired simulator on The Platform through this interface. Instead of the intensity, position, and progression of the earthquake being processed by the DM, these (possibly among other) aspects would be provided to the DM through a software layer which would inform the DM of the required data. This should be a means of programatically inject information from an external simulator into the DM.

The focal points of the developed thesis work can be exposed in the form of the following research questions:

- Is it possible to create an environmental disturbances manager, and effectively integrate it within the existing platform?
- Is it feasible to introduce a generic interface for external disaster simulators into the system?

The first question represents the implementation of the DM on the platform. "Effectively" is used as a key word, since even if the DM is implemented with a wide array of useful functionalities, if it significantly impacts the real-time performance of the platform, it would not be of use. The second question regards the aforementioned secondary objective, with "feasible" being a meaningful term - because if the implementation were to generate a delay in the processing time to the degree of impeding real-time performance, it would also not be useful.

The Platform's performance is considered significantly negatively impacted if it becomes unable to run not only real-time simulations, but also simulations at double or even four times the real-time simulation speed. On Chapter 5, several methods were used in order to validate the performance with a running DM.

Many components of the DM would be sufficiently complex as to warrant multiple dissertation topics by themselves, such as the time evolution of the disturbances, their spacial dispersion, how and when they should occur, and so on. Taking this into consideration, the project at hand will not provide an accurate representation for each possible disturbance type; instead, the aim is to provide a Disturbances Manager as proof of concept - that it is possible to integrate such simulations into the Platform. The inclusion of an optional external disturbance-specific simulator aims to ameliorate the limitations derived from the genericity of this approach.

With the exception of the external co-simulator (more details on Final Work Overview), all the planned objectives were obtained. Validation corresponding to the main aspects of the implemented functionalities was also performed, as detailed on Testing & Results.

1.3 Document Structure

The remaining sections of this document are structured as follows:

Chapter 2 consists of a summary of the state of the art and key concepts required to fully understand this dissertation's subject matter, including what is considered a disturbance and a disaster simulator, in the context of the project at hand; what is a flight simulator; and what composes an architecture with co-simulators.

Chapter 3 exposes the chosen methodology regarding the developed work, including planning, risk management, the chosen strategy, and technological choices - such as programming languages.

Chapter 4 lists how the solution fits into the existing architecture, namely by expanding upon The Platform's enhanced architecture, the Disturbance Description Language (DDL) and the Disturbance Database. It also includes a description of the Disturbances Manager's functionalities. The chapter also contains details regarding the implemented Disturbances Manager, including the initial prototype, the vehicles' sensor readings and communication with the corresponding elements of the Platform's architecture.

The 5th chapter includes information on the creation of the performed tests, and the matching results.

Finally, chapter 6 consists of the conclusions reached throughout the project and includes some possibilities for future work

Chapter 2

State of the Art

This chapter consists of a listing of the key concepts involved in the project, as well as a summary of the researched state of the art on the related subjects. It starts with the adopted definition for the term *disturbance*, followed by a listing and comparison of several types of environmental disturbance-related simulators. An interesting researched architecture for systems with co-simulators is also presented.

2.1 Disturbances - Environmental and Beyond

An **Environmental Disaster** is an event which negatively impacts the natural environment, often with a high and/or long-lasting impact on the ecosystem in question. Occasionally, only occurrences resulting from human activity are considered environmental disasters - using the concept of **natural disaster** for incidents which occur without human intervention. Throughout this document, the expression **Environmental Disturbance** will encompass both concepts.

Throughout the research process, multiple sources were useful in providing information and statistics on specific environmental disasters:

1. Government entities from the United States of America proved to be a valuable source of data. Not only on finding the most serious [NOAA, 2017b] (in terms of lives taken and financial damage) but also on finding statistics on the increase in yearly occurrences [NOAA, 2017a];

2. Certain websites on the topic of disaster preparedness were also useful since they often include listings on not only natural but also man-made disasters [IEDC, 2016];

3. An Indian government entity was also a good source on not only the types of existing disturbances and their details, but also on their potential impact: with examples of the most noteworthy disasters [NIDM, 2014];

4. A paper was found on the future prospects for disturbances [Quarantelli, 1992] which proved to be rather insightful on the subject of what can be expected, and what can be done regarding future disasters;

5. How should one study environmental disasters? At the early stages of the research work at hand, there was some difficulty in researching and quantifying knowledge on the subject. A research article provided valuable input on the subject [Drabek, 1970], and brought upon the idea of introducing the Seriousness Degree on the specification of disturbances, and also on defining the level of abstraction which was adopted on the listing and specification of the disturbances chosen.

Furthermore, considering that there is no restriction on what a Disturbance may be, it was decided to include non-environmental situations to the list of cases that are intended to be simulated - such as car and train accidents, missing people and oil spills. Onward, the term **Disturbance** will encompass all those listed in Tables 2.1 through 2.4). The following paragraphs include a description on the rationale behind the grouping of the disturbances, as well as some details on them.

The Land Disturbance category encompasses all the considered disturbances which occur on land, both literally (landslides, earthquakes and volcanoes) and not (e.g. floods, accidents). These are listed on Table 2.1.

Table 2.1: Land Disturbances

Name	Origin	Sensor Types	Seriousness Degree	Growth Rate
Landslide	Natural	CameraIntensity; Microphone	yes	yes
Earthquake	Natural	CameraIntensity; Microphone	yes	yes
Flood	Natural	CameraIntensity	yes	yes
Volcano	Natural	CameraIntensity; IRCameraIntensity; Temperature; Microphone	yes	yes
Car Accident	ManMade	CameraBinary	yes	no
Train Accident	ManMade	CameraBinary	yes	no
Fleeing Person	ManMade	CameraBinary; Microphone	yes	no
Lost Animal	ManMade	CameraBinary; Microphone	yes	no
Forest Fire	Any	CameraIntensity; IRCameraIntensity; CO_2; Temperature	yes	yes
Building Fire	ManMade	CameraIntensity; IRCameraIntensity; Temperature; CO_2	yes	yes

The Sea Disturbance category aggregates all the selected disturbances which take place at sea. From tsunami to chemical spills, many were considered relevant to include on The Platform. These are listed on Table 2.1.

The Storm Disturbance category accumulates the chosen disturbances which are related to storms that were not listed under Land or Sea disturbances, since they can occur both on land and at sea. The category was detailed on Table 2.3.

The General Disturbances category includes those which would not be considered good fits for the previous categories (Land, Sea or Storm). The General Disturbances are listed in Table 2.4.

The sensors indicated in tables 2.1, 2.2, 2.3 and 2.4 are the following:

Table 2.2: Sea Disturbances

Name	Origin	Sensor Types	Seriousness Degree	Growth Rate
Strong Waves	Natural	CameraIntensity; Microphone	yes	yes
Tsunami	Natural	CameraIntensity; Microphone	yes	yes
Oil Spill	ManMade	CameraIntensity	yes	no
Chemical Spill	ManMade	CameraIntensity ; Chemical	yes	yes
Boat Accident	ManMade	CameraBinary	yes	no
Hydrothermal Vent	Natural	CameraIntensity; Temperature; Microphone	yes	yes
Submarine Volcano	Natural	CameraIntensity; Temperature; Chemical; Microphone	yes	yes
Schooling Fish[1]	Natural	CameraIntensity	yes	yes

[1] A large aggregation of fish, relevant to submarines and boats

Table 2.3: Storm Disturbances

Name	Origin	Sensor Types	Seriousness Degree	Growth Rate
Heavy Rain	Natural	CameraIntensity; HumiditySensor; AirPressure; WindSpeed; Microphone	yes	yes
Thunderstorm	Natural	CameraIntensity; HumiditySensor; AirPressure WindSpeed; ElectricField; MagneticField; Microphone	yes	yes
Hurricane	Natural	CameraIntensity; AirPressure; WindSpeed; Microphone	yes	yes

Table 2.4: General Disturbances

Name	Origin	Sensor Types	Seriousness Degree	Growth Rate
Bug Infestation	Natural	CameraIntensity	yes	yes
Extreme Temperature	Natural	Temperature	yes	yes
Cold Wave	Natural	Temperature	yes	yes
Heat Wave	Natural	Temperature	yes	yes
Chemical Contamination	ManMade	Chemical	yes	yes
Chemical Cloud	ManMade	Chemical	yes	yes
Radiation	ManMade	GeigerCounter	yes	yes
Under Fire	ManMade	CameraIntensity; Microphone	yes	yes
Buildings In Ruins	ManMade	CameraBinary	yes	no

- ***Airpressure*** - A sensor for air pressure, which typically displays readings in the *atm* unit;
- ***CameraBinary*** - A camera, which reads *true* or *false*, according to whether the disturbance in question can be observed;
- ***CameraIntensity*** - A camera, which reads an observed disaster intensity;
- ***Chemical*** - A generic chemical sensor, which may convey readings in a specified unit;
- **CO_2** - A Carbon Dioxide sensor, which reads the concentration of the chemical in the air;
- ***ElectricField*** - Measures the intensity of affecting electric fields, typically in newtons per coulomb (N/C);
- ***GeigerCounter*** - Indicates radiation intensity, measured in counts per minute (CPM);
- ***HumiditySensor*** - Measures the humidity of the air, typically indicated in grams of water vapor per cubic meter of air;
- ***IRCameraIntensity*** - An infrared camera, which reads an observed disaster intensity;
- ***MagneticField*** - Measures the intensity of affecting electric fields, typically in Teslas (T);
- ***Microphone*** - Reads the surrounding sound intensity;
- ***Temperature*** - A thermometer, which reads given ambient temperature;
- ***WindSpeed*** - Obtains the speed of the wind.

The *seriousnessDegreee* indicates whether a degree of seriousness is applicable to the disturbance in question, and similarly the *growthRate* indicates whether a rate of growth is applicable for the disturbance at hand (e.g. a "missing person" can not grow in intensity but can grow in seriousness; however a chemical spill may increase in both intensity, i.e. how much chemical is being spilled, and seriousness, given the region to which it may be moving).

2.1.1 Environmental Disturbance Simulators

At the initial stage of the research regarding Environmental Disturbance Simulators, the scope was too broad, as the search was performed in order to find simulators for any type of disturbance.

Many of the encountered applications were too simplistic, e.g. taking form as children's informative games such as *Hurricane Simulation*[1]; while others were far too complex to consider, resulting in more computationally expensive simulations, and all the intrinsic detail being of no potential use to the DM.

Before deciding on narrowing the scope of the search (see below: Forest Fire Simulators), three simulators stood out among the research results - these are detailed and compared in the following paragraphs, as well as in Table 2.5.

[1]Available at `http://scijinks.gov/hurricane-simulation/`

Virtual Quake[2] is a simulation program which models the fault[3] system in California. Several geological details regarding the region's characteristics are used to simulate an earthquake. Virtual Quake claims a design which allows for the fast execution of the thousands of small events involved in the calculations. Its execution results in a detailed *dataset*, which may be used to study several aspects of the simulation. The application offers the option of outputting the simulation results in an HDF5 file[4], allowing for posterior processing and analysis of the data.

F5Data[5] is mainly a weather simulator and forecasting program. It simulates numerous weather aspects, from sea surface temperatures to wind flow. Despite it's wide array of features, including a hurricane component, the program was not considered a strong applicant for the planned co-simulation for its merely superficial approach to environmental disturbances, and only offering the option to save data as an image. It is also not open-source software an does not provide a good manual or documentation.

NERIES ELER V3.1[6] is perhaps the most complete of the gathered simulation programs. It estimates the propagation of the seismic waves through the earth from a given epicenter and magnitude value, offering the option of including additional geological data regarding the region at hand to improve the accuracy of the results. The tool also computes an estimation of building and human physical damage on the affected region. It was developed using the *MATLAB*[7] programming environment, and uses *MATLAB* to display the results, but also offers the option of exporting data to popular *GIS* programs[8]. Since this tool does not export the simulation results to a common format such as XML (Extensible Markup Language) or JSON (JavaScript Object Notation), the mentioned *GIS* exporting functionality requires further investigation as to assert the feasibility of utilizing *ELER* as a co-simulator on The Platform.

Table 2.5 provides a summary of the mentioned simulators, namely the respective names, whether they are free, open-source and the perceived quality of the documentation (*1* being the lowest, including only a few code examples; *2* as the intermediate level, with better code examples or tutorial videos; and *3* as the highest quality, corresponding to a detailed and well-written documentation).

Table 2.5: Environmental Disturbance APIs Comparison

Name	Free	Open-Source	Documentation (1-3)
Virtual Quake	Yes	Yes	2
F5Data	Yes	No	1
NERIES ELER	Yes	No	3

[2]Available at `https://geodynamics.org/cig/software/vq/`

[3]A fault zone is a zone where small earth dislocations have occurred [Brodie et al., 2007]

[4]Hierarchical Data Format 5 - a file format designed for storage of large amounts of data

[5]Available at `http://www.f5data.com/`

[6]Available at `http://www.koeri.boun.edu.tr/Haberler/NERIES%20ELER%20V3.1_6_176.depmuh`

[7]Available at `https://www.mathworks.com/products/matlab.html`

[8]"A geographic information system (GIS) is a computer system for capturing, storing, checking, and displaying data related to positions on Earth's surface. GIS can show many different kinds of data on one map, such as streets, buildings, and vegetation." [NG, 2017]

2.1.2 Forest Fire Simulators

Computer simulations of forest fires have been performed for over 40 years [Stevenson et al., 1975]. These simulations can be highly complex, and many mathematical methods are applied in order to perform the matching calculations, from probability spread calculations [Ivanilova, 1985] to multiplex networks [Buscarino et al., 2015]. These types of simulations are widely used across several entities, such as the U.S. Forest Service and the U.S. National Park Service [USDA, 2015].

Considering the wide variety of existing environmental disturbances, many corresponding software simulators exist. It was decided on narrowing the scope of this component of the research to a specific type of disturbance.

Portugal has an immense risk of forest fires [NICIF and Lourenço, 1994], mainly on account of its geographic characteristics. As such, tens of thousands of hectares of forest burn on a yearly basis [MAMAOT, 2015]. Taking these facts into consideration, the decision was made to focus the research regarding environmental disaster simulators to simulators of fires - specifically forest fires. Another advantage of the simulation of fires, is that they can be very representative of a simulator's capacities: from simple aspects such as higher temperatures to more complex cases such as the spreading patterns of smoke.

FlamMap[9] is program which maps and computes fire behavior, including spread rate and intensity. It is developed by U.S. Forest Service, which belongs to the United State Department of Agriculture; free to use, yet not open source. The software is not a complete fire growth simulation model, focusing on calculating fire behavior characteristics for specific sets of environmental conditions. The functional principle of the program is the creation of raster maps of the terrain, with each pixel potentially containing certain fire behavior details and other aspects such as wind speeds - this is performed over a FARSITE (detailed below) landscape [A. Finney, 2006]. These raster maps can be visualised within FlamMap or exported in other formats. It does not provide an API.

FARSITE[10] is similar to FlamMap, and also developed by the U.S. Forest Service, yet focuses on the behavior of fires throughout long time periods and across multiple terrain types; using multiple popular fire behavior models, such as Rothermel's [Wells, 2008] surface fire spread model. FARSITE aggregates existing models of surface fire using a vector propagation technique [Finney and Station-Ogden, 1998]. The software is used across many institutions in the United States of America, such as the National Park Service[USDA, 2015]. It does not provide an API.

ForeFire[11] is an open-source simulation engine designed for large scale fire simulation which takes standard landscape data as input. It includes numerous bindings for different programming languages (such as Java and Python), as to provide easy integration in other projects. Its goal is to provide a set of open simulation tools and a Javascript API. ForeFire is based on the propagation speed model of Balbi et al. [Henri Balbi et al., 2009][Filippi et al., 2011].

[9] Available at `https://www.firelab.org/project/flammap`
[10] Available at `https://www.firelab.org/project/farsite`
[11] Available at `http://forefire.univ-corse.fr/sim/dev`

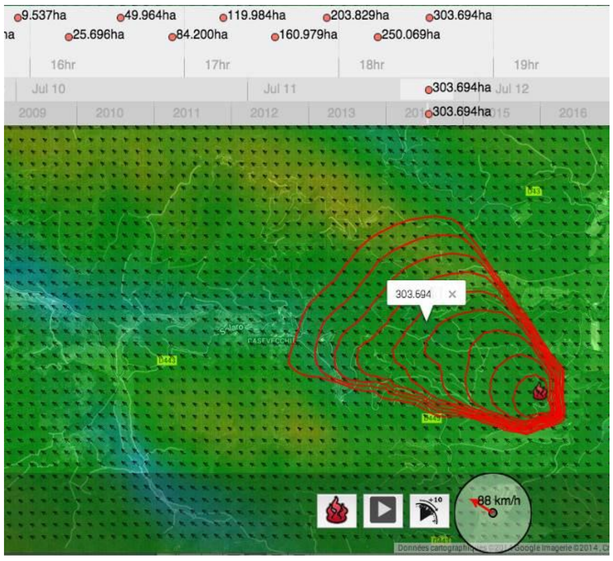


Figure 2.1: ForeFire

Capaware[12] is an open-source framework which provides real-time forecasting of the evolution of fires, developed in the Technological Institute of the Canary Islands. The program is focused on data visualization, and offers a free API as to facilitate the development of new applications around it. However, the corresponding documentation appears lacking.

Wildfire Analyst[13] provides real-time (i.e. one second of simulation time corresponds to one second in what would happen in real life) simulation and analysis of the spreading of wildfire (fire within an region of combustible vegetation). Provides a wide range of output data and a fully featured API. Regarding the set of features, Wildfire Analyst appears to be one of the best options encountered. However, it is not free to use.

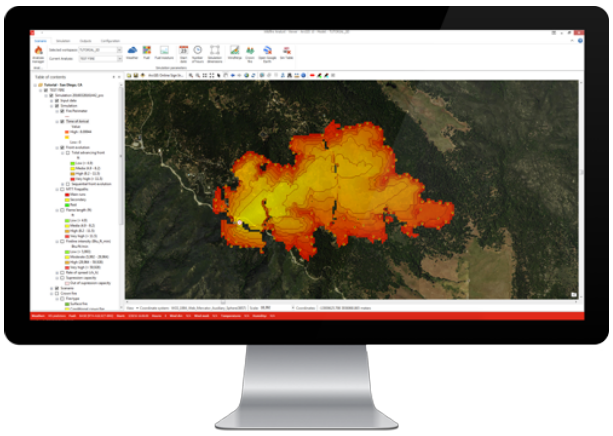


Figure 2.2: Wildfire Analyst

Table 2.8 provides a summary of the mentioned fire simulators, namely the respective names, whether they are free, open-source and the perceived quality of the documentation (*1* being the

[12]Available at http://www.capaware.org

[13]Available at http://wildfireanalyst.com/

lowest, including only a few code examples; *2* as the intermediate level, with better code examples or tutorial videos; and *3* as the highest quality, corresponding to a detailed and well-written documentation).

Table 2.6: Fire Simulators Comparison

Name	Free	Open-Source	Provides API	Documentation (1-3)
Capaware	Yes	Yes	Yes	2
Farsite	Yes	No	No	1
Forefire	Yes	No	Yes	2
FlamMap	Yes	No	No	1
Wildfire Analyst	No	No	Yes	3

Pondering over the considered platform, Capaware was considered the best candidate for integration with The Platform. However lacking in documentation, being the only one both free and providing an API makes it the preferred choice. If there were the possibility of obtaining a free or academic license for Wildfire Analyst, it would also be a good choice - not being open-source, it could make up for it by the quality of the documentation provided.

2.1.3 Meteorology APIs

Since the recent boom of open data [Gewin, 2016], a wide array of weather-related APIs also became freely available [Santos, 2012]. The original intent of using a co-simulator was to allow a more lifelike representation of environmental disturbances, by providing them with more accurate behaviours, not limited by the standardized models within The Platform. Given that the result of the Environmental Disturbance Simulators research returned a relatively reduced number of simulators, it was considered relevant to investigate Meteorology API's as to use real disturbance data instead of simulated data.

Real-life data is, by definition, more reliable than any simulation. Therefore the data from one of the gathered APIs could be utilized to provide a specific disturbance's behaviour and evolution, emulating a simulator by processing and transferring the information to The Platform in real time, as if the registered event was happening again. However this would not invalidate the need for a simulator, given that one might wish to see the results of a specific disturbance had, for example, the weather conditions been different. One might also wish to change other parameters of a simulation, in order to simulate disturbances like none that ever existed - i.e. with no real-life data available on it.

Multiple APIs were found which not only provide the typical weather related information (e.g. temperature, wind and humidity) but also have a component regarding environmental disturbances. Most of them are free to use - however subject to daily request limits - and not open-source, however there are some exceptions.

Given that the studied APIs are considerably similar, they have been included in Table 2.7 as to provide a simpler overview on all. The noted differentiation factor for each API was also included.

Table 2.7: Meteorology APIs Comparison

Name	**Free**	**Open-Source**	**Documentation (1-3)**
Accuweather[a]	Free[b]	No	3
OpenWeatherMap[c]	Free[d]	No	3
Baron Velocity Weather[e]	Free	No	2
Aerisweather[f]	Free[g]	No	2
USGS Earthquake[h]	Free	No	3
Dark Sky[i]	Free[j]	No	3
Weather Underground[k]	Free[l]	No	3
NOAA[m]	Free	No	3

[a]Accuweather's API is available at https://apidev.accuweather.com/developers
[b]Free to use up to the set daily/monthly request limit
[c]The OpenWeatherMap API is available at http://www.openweathermap.com/current
[d]See footnote (b)
[e]The Baron Velocity Weather API is available at https://www.velocityweather.com/products/weather-data-api
[f]The Aerisweather API is available at http://www.aerisweather.com/support/docs/api/
[g]See footnote (b)
[h]The United States Geological Survey Earthquake Catalog API is available at https://earthquake.usgs.gov/fdsnws/event/1/
[i]The Dark Sky API is available at https://darksky.net/dev/
[j]See footnote (b)
[k]The Weather Underground API is available at https://www.wunderground.com/weather/api/
[l]See footnote (b)
[m]National Oceanic and Atmospheric Administration's Weather and Climate Toolkit is available at https://www.ncdc.noaa.gov/wct/

- Accuweather was found to be highly regarded, given it's accuracy;
- OpenWeatherMap includes UV index, air pollution, and tornadoes;
- Baron Velocity Weather's provides tornado tracking;
- Aerisweather supports thunderstorms and volcanic ash (harmful for airplanes [USGS, 2016]).
- USGS works with both real-time and archived earthquakes;
- Dark Sky supports real-time disaster warnings, which could be used to trigger disasters in-simulation. However only includes a superficial disaster component;
- Weather Underground supports real-time disaster warnings, which could be used to trigger disasters in-simulation. Provides movement data of active hurricanes;
- NOAA allows for the exporting of data in a wide array of data formats. Supports thunderstorms, tornadoes and blizzards.

After considering all the mentioned Meteorology APIs, Weather Underground's API was deemed the best option therein - being free to use, providing thorough documentation and supporting real-time disaster warnings. Ergo, the API should be a good contender to simulate the role of co-simulator.

2.1.4 Disturbance Representation in FSX

Given the popularity of the flight simulator, a wide community was formed around it. With thousands of active members, many tools and add-ons were created for the game.

While not the focus of the dissertation at hand, having some kind of in-game representation of the disturbances would be an interesting and visually appealing addition to the project. Multiple tools which enable the representation of various weather effects in the game were studied as to find an acceptable way of achieving the in-game representation of environmental disturbances.

Besides the potential in achieving a visually appealing solution, it could also become possible to have the vehicles obtain specific readings from the simulator itself - thereby reducing the need for processing and information transferring from outside FSX.

2.1.4.1 Mission Creation

The SDK (Source Development Kit) for Flight Simulator X includes a Mission Creation component [Microsoft, 2008a]. By creating missions, through the editing of XML files or via a graphical interface, one can set up various activities for the player - which can be associated via triggers to in-game visual effects, e.g. smoke or fire, which could be used for the desired purpose.

However versatile in the possibilities for in-game disturbance representation, FSX's missions pose a great disadvantage: they require compiling and posterior loading at the start of the simulation; this does not agree with the objective of deploying disturbances mid-simulation. Thereby, the Mission Creator was set aside. However, as an alternative, the Mission Creator could be utilized by having the missions created and compiled at the start of the simulation. This would provide the desired results, with the restriction of the creation of the disturbances only at the start of the simulation.

2.1.4.2 Special Effects Tool

FSX's SDK also includes a Special Effects Tool [Microsoft, 2008c]. The major advantage of the tool in the context of representing disturbances on-screen would be that it allows for a special effect (e.g. fire) to have specific movements and dispersion patterns, as is expected of the DM. However, the special effects can also only be used through generating the file, compiling and loading it before the simulation starts. This is a restriction of the game's core engine, therefore unavoidable; as such, the Special Effects Tool was also put aside. As with the Mission Creation, one could circumvent this limitation by specifying the disturbances, generating the matching Special Effects files and compiling them before the start of the simulation.

2.1.4.3 SimConnect

SimConnect [Microsoft, 2008b] is the core of FSX's SDK, allowing several types of information exchange between the Simulator and external programs. While not offering many options regarding visual effects, SimConnect's foremost advantage is the fact that ir runs in real-time during the simulation.

SimConnect's features were studied with focus on finding a way to make use of it as to show an in-game indication of environmental disturbances. Through combining its ability of loading aircraft models and the possibility of activating specific events within the aircraft (e.g. turning on lights or retracting/expanding landing gear) the researcher arrived upon the possibility of deploying an airplane at the center of the disturbances, as to later enable their smoke emission engine - signaling, for example, a fire at the same location. Considering that the deployed airplane itself was just a means to achieving the possibility of showing smoke, and as to not add unnecessary visual elements to the simulation, the aircraft chosen to show the smoke can be made invisible through the editing of aircraft configuration files. Later, it was discovered that the aircraft's emission can be changed from smoke to a different effect such as water, fire, or even fireworks, which allows for a more diverse in-game visual representation of disturbances.

The mentioned limitation of SimConnect was surpassed in SimConnect Version 4 for Lockheed Martin's Prepar3D simulation software [Martin, 2017], which allows the creation of Special Effects in runtime. The latest SimConnect is unfortunately not available for FSX.

2.1.4.4 Weather Engines

In the context of FSX, weather engines focus on enhancing in-game weather effects: most commonly through not only enhancing the visual fidelity of clouds and other aspects, but also by changing the in-game weather to match what is happening in real-life. In the search for maximum in-game visual fidelity, many weather engines were developed for the game by various parties. Through research, many appeared interesting and potentially useful for the project.

Amongst all the encountered weather engines, the following were the most noteworthy on account of their feature sets

- ***FSXWX***[14]
- ***FSrealWX***[15]
- ***AS16***[16]
- ***OpusFSI***[17]
- ***Weather Architect***[18]

All of which, being weather engines, are focused on injecting weather into the simulation: not only with more accurate behaviour than default (as in matching real-world weather in terms of real-time updates and location), but also - and above all - looking more lifelike. Unfortunately,

[14]Available at http://www.plane-pics.de/fsxwx/home.htm
[15]Available at http://www.fsrealwx.net/
[16]Available at http://hifisimtech.com/as16/
[17]Available at http://www.opussoftware.co.uk/opusfsi.htm
[18]Available at http://www.rexsimulations.com/weatherarchitect.html

only AS16 provides an API, and even that does not allow for the injection of weather occurrences in real time onto the simulation. Besides, all these weather engines rely on stopping the simulation for a significant amount of time as to process all the data at hand. It was during this research that game engine's limitation mentioned in the Special Effects section above was discovered.

2.2 Architectures with Co-Simulators - HLA

In order to achieve a better understanding of what is typically involved in and required for an architecture with a co-simulator, it was decided to research the HLA architecture.

The High Level Architecture (HLA) is an IEEE standard [IEE, 2010] to specify distributed computer simulation systems. Unfortunately, the official IEEE document was not freely available at the time of research, so alternate documents [Reid and Powers, 2000][Dahmann et al., 1998] were utilized in order to comprehend the HLA.

One of the core purposes of the HLA is "the continuous reuse and interoperation of simulations" [Dahmann et al., 1998]. The main advantage of using an HLA is that one can "reduce software costs by facilitating the reuse of simulation components and by providing a runtime infrastructure to manage the simulations." [Reid and Powers, 2000] In summary, the Architecture allows the user to save resources by managing the simulations through a specific infrastructure - which allows for the communication and synchronization of actions between different simulations.

HLA has a specific terminology:

- A **Federate** is an HLA-compliant simulation entity;
- A **Federation** is a set of connected simulations;
- An **Object** is a set of data transferred between simulations. Conceptually, an Object is an entity being modeled by the simulation;
- An **Interaction** is a message that any Federate within the execution can transfer or receive;
- The **Runtime Infrastructure** is the simulation itself, i.e. the core software;
- An **Object Model Template** is what defines the format for the data being transferred during the simulations.

HLA's rules describe how the Federates and Federations act, and has the goal of allowing simulation applications to be structured as a set of smaller simulations - which allows for the re-utilization of specific "sub-simulation" results, therefore saving resources.

2.3 Summary/Conclusions

Regarding the Disaster Simulators and the meteorology API, it is considered that the acquired information will be of great value to the development of the intended solution. With a wide array of options, there are many fail-safe alternatives - in case one or more of the mentioned tools somehow does not work as expected during the implementation stage.

Regarding the in-game representation of the disturbances, along with the decided focus on simulating forest fires, the three listed options were analyzed as to elect the best option for the implementation, weighing their pros and cons. It was decided to use the SimConnect approach - by combining the smoke (or other effects) emission from an invisible airplane spawned at the center of the disturbances. This provides a simple and time-effective way of achieving the desired result of a visual representation in FSX.

On the subject of fire simulators and meteorology APIs, many interesting options were analyzed, with Capaware and Weather Underground's API being regarded as the prime candidates for integration in The Platform.

As to the HLA architecture, it proved to be an interesting approach to the structuring and organization of simulation systems, and will be taken into consideration throughout the development of the solution and its integration into The Platform.

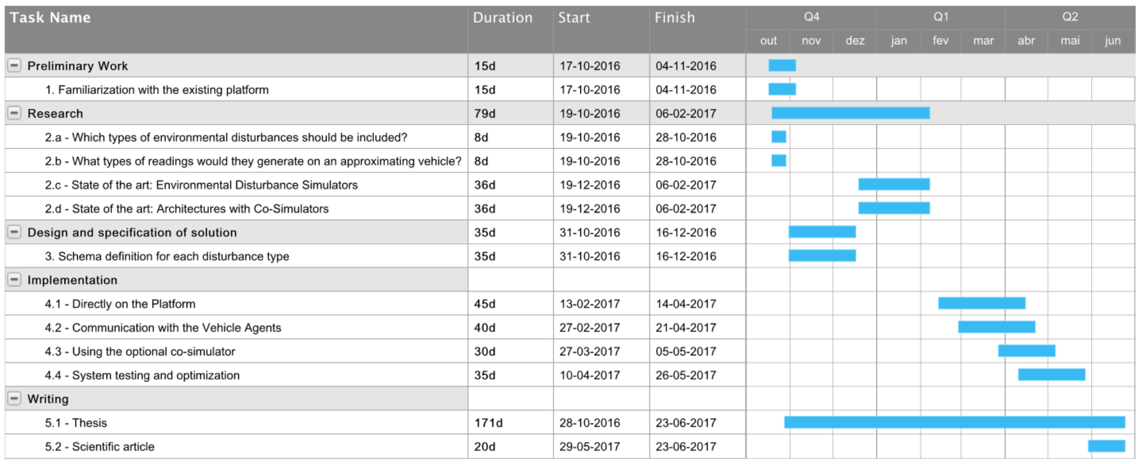
Chapter 3

Planning & Methodology

This chapter introduces the preliminary decisions related to all the planning, practical and research work at hand. First, each of the main work stages is elaborated upon; second, the risk management is detailed; third, the planned methodology is briefly explained; and finally, the technological choices related to the implementation are described.

3.1 Work Plan

The following section of the document includes the summary of the work plan elaborated in order to specify all the time and task management regarding the project at hand. Each of the stages on the plan is also included in the Gantt Diagram on Fig. 3.1 - which includes the start and end dates for each stage.



Task Name	Duration	Start	Finish
Preliminary Work	15d	17-10-2016	04-11-2016
1. Familiarization with the existing platform	15d	17-10-2016	04-11-2016
Research	79d	19-10-2016	06-02-2017
2.a - Which types of environmental disturbances should be included?	8d	19-10-2016	28-10-2016
2.b - What types of readings would they generate on an approximating vehicle?	8d	19-10-2016	28-10-2016
2.c - State of the art: Environmental Disturbance Simulators	36d	19-12-2016	06-02-2017
2.d - State of the art: Architectures with Co-Simulators	36d	19-12-2016	06-02-2017
Design and specification of solution	35d	31-10-2016	16-12-2016
3. Schema definition for each disturbance type	35d	31-10-2016	16-12-2016
Implementation			
4.1 - Directly on the Platform	45d	13-02-2017	14-04-2017
4.2 - Communication with the Vehicle Agents	40d	27-02-2017	21-04-2017
4.3 - Using the optional co-simulator	30d	27-03-2017	05-05-2017
4.4 - System testing and optimization	35d	10-04-2017	26-05-2017
Writing			
5.1 - Thesis	171d	28-10-2016	23-06-2017
5.2 - Scientific article	20d	29-05-2017	23-06-2017

Figure 3.1: Work Plan Gantt Diagram

1. Familiarization with the existing platform - the goal of this task is to become acquainted with The Platform. This involves the necessary setup, running it, performing some experiments

and analyzing the existing code. Since the Platform is of a considerable dimension, this phase is essential to the start of the project

2. Research
   (a) Which types of environmental disturbances should be included? At this stage, a list should be created to provide a summary of all the disturbances to be simulated. As well as a list of the main sensors necessary to acquire readings related to the disturbances at hand;
   (b) What types of readings would they generate on an approximating vehicle? Following the previously mentioned lists, each disturbance should be assigned at least one reading - to be generated on a specific sensor;
   (c) State of the art: Environmental Disturbance Simulators. At this point, the most relevant, potentially useful, and most modern simulators of environmental disturbances should be researched and catalogued, specifically to help make the decision of the external co-simulator which should become an optional element of the Platform;
   (d) State of the art: Architectures with Co-Simulators. Software architectures incorporating co-simulators should be analyzed as to acquire the knowledge necessary for the development of the project: it will allow for the researcher to become familiarized with the industry standards; to know the existing alternatives; and to avoid building an architecture from the start when existing and proven architectures exist - which can be considered as good examples.
3. Design and specification of solution
   (a) Schema definition for the disturbances. Based upon the information gathered on the types of disturbances and related readings, the Platform's existing XML Schema for disturbances should be altered as to allow for the inclusion of the required elements not previously specified.
4. Implementation
   (a) Directly on The Platform - the first stage of implementation is related to the starting of the DM itself. After the updating of the aforementioned schemas, the related C# class files should be updated as to allow for the correct reading and saving of XML files, as well as their de-serialization into Objects. The DM should be launched through the Control Panel and handle the loaded disturbances by managing their location, deployment, evolution, and most importantly sending the sensor readings to the vehicles;
   (b) Communication with the Vehicle Agents - as mentioned before, the DM will communicate with the vehicles to provide them with sensor reading information. As a means to achieve this communication, AgentService (introduced in 3.5.3) will be used. A message structure should be specified; the messages should be generated, sent, received and processed by each agent - which save a log file with the received readings;

(c) Using the optional co-simulator - the interface between the DM and an external disturbance simulator should be developed as to allow the DM to process specific disturbances more accurately by gathering instructions on their behavior (i.e. intensity and dispersion patterns, among other aspects) from the external simulator. The interface should be sufficiently generic as to allow for easy adaptation of any simulator which provides an API;

(d) System testing and optimization - this stage, which occurs after the development itself, concerns the optimization and testing of the developed components. A set of tests should be specified and performed on The Platform, with the results being registered and analyzed. Throughout this stage, if any functionality of the software is found to be excessively slow, it should be altered as to maximize performance.

5. Writing

   (a) Thesis - the thesis, despite being the final aspect of the work to be delivered, should be advanced throughout the entire work period. This allows for an iterative approach, with feedback from the supervisor and constant updating;

   (b) Scientific Article - by the end of the work period, a scientific article is to be written, summarizing the performed work and achieved results, with the goal of being submitted and later presented on a conference.

This section, along with Fig. 3.1, provides information on the work plan which was developed at the start of the dissertation planing stage. Before mentioning the risk management aspects regarding the project, the actual - as was completed - work schedule is included in the next subsection.

3.2 Final Work Overview

In this section, all the points of the work plan will be scanned, providing details on how the performed tasks matched the work schedule; as well as mentioning the related Risk Management items.

1. **Familiarization with the existing platform** - This stage of the work plan was executed as expected, with no significant difficulties other than fully grasping the intricacies of each of the architecture's components. Being a considerably large project, some difficulty was expected; however, with the aid of the supervisor in clarifying some of the components' functionalities, the familiarization was performed successfully. As a result, the first potential risk (i.e. the first entry in the Risk Management subsection below) was successfully avoided.

2. **Research**

   (a) **Which types of environmental disturbances should be included and what types of readings would they generate on an approximating vehicle?** - As mentioned in section 2.1, through research and common knowledge, four lists of disturbances were created for the following categories: Land, Storm and General. Work went as expected, and while gathering each disturbance type, the corresponding possible origin (Natural or Man-Made) was specified, as well as the matching vehicle sensor types. Lastly, for each of the listed disturbances, it was decided whether they should have an associated *seriousnessDegree* and a *growthRate* - for example, it is obvious that a forest fire should have a specific growth rate, while it does not make sense to have a growth rate for a lost person (the person is either lost or not lost, it's not considered that someone could become somehow "more" lost). This work stage took the expected amount of time, and was performed without difficulties.

(b) **State of the art: Environmental Disturbance Simulators** - As described in Chapter 2, at first, the spectrum of the research was excessively wide. This resulted in a fair amount of time being spent on researching simulators for various types of disturbances before the scope was narrowed to fire simulators. The possibility for this delay was considered in the second item of the Risk Management subsection - as planned, the work schedule was appropriately restructured and advice was taken from the supervisor.

(c) **State of the art: Additional Work** - While not considered in the original work plan, as it was considered an interesting addition to the project, several options to obtain in-game representation of the disturbances' effects were analyzed (subsection 2.1.4). Given the vast amount of options encountered, a significant amount of time was allocated to this task. This relates to the third entry in the Risk Management section - the unveiling of unexpected tasks, and was handled accordingly.

(d) **State of the art: Architectures with Co-Simulators** - This section of the research was completed without delays. The supervisor suggested the research of the HLA, which was investigated.

3. **Design and specification of solution**

   (a) Schema definition for the disturbances - This stage was performed in a small amount of time; however the initial specification was not final, as a few additions were performed during the DM implementation stage, when it became apparent that they were necessary. Therefore this stage of the project was performed somewhat iteratively, throughout more time than what was initially expected, however did not exceed the planned time.

4. **Implementation**

   (a) **Directly on The Platform** - As planned, this was the most prolonged stage of the entire project. Due to multiple unexpected obstacles amid the development, the mitigation strategy for the second risk management item was utilized - thereby re-allocating the available work time and taking advice from the supervisor.

(b) **Communication with the Vehicle Agents** - Considering that other components of The Platform (i.e. the ATC agent) perform comparable message exchanges with Vehicle Agents, a significant amount of the existing code could be taken as an example; as such, by performing a few alterations and additions, the communication between the DM and the Vehicle Agents was achieved.

(c) System testing and optimization.

5. **Writing**

(a) **Thesis** - The dissertation should be written iteratively from the start. This should allow for an iterative approach, adding content as the implementation are completed.

(b) **Scientific Article** - The planned scientific article is currently being developed, and will be completed as to be sumitted in a relevant conference.

Considering the risk mitigation strategies, namely the allocation of additional time to the tasks above, a fair amount of available work time was reduced. Therefore, it was decided to exclude the optional co-simulator from the development plan. While not the least meaningful element of the project (which would be the visual representation of the disturbances), it was a stage which could be removed without affecting the work performed thus far.

3.3 Risk Management

The following represent the most significant risks regarding the project at hand, as well as the corresponding risk mitigation strategies.

- Risk 1: failure to grasp the scope of the Platform. Mitigation: early and frequent meetings with the supervisor to clear any existing questions; allocation of sufficient time to explore the Platform and read about it.

- Risk 2: one or more of the implementation stages sustaining significant delay, or simply taking longer than planned. Mitigation: re-structuring of the work plan; allocating more time to the task at hand; taking advice from the supervisor.

- Risk 3: the necessity to perform unexpected tasks. Mitigation: similarly to the previous point, the re-structuring of the work plan and allocated times, as well as considering input from the supervisor.

The listed risks have been assigned impact and probability values in Table 3.1.

3.4 Work Strategy

Throughout the development of the remaining aspects of the project, indicated in the Work Plan, the following iterative strategy was adopted: individual work, followed by a meeting with the Supervisor. This was repeated on a weekly basis throughout the entire development process.

Table 3.1: Risk Table

		Probability		
		Low	Medium	High
Impact	High			
	Medium	R1		R2
	Low			R3

3.5 Technologies and Platforms

In the following section, the main technologies related to the work at hand will be described. Considering that the project in question fits within an existing system, the technologies mentioned had already been established, therefore they will not be significantly compared to others.

3.5.1 The Platform

The Platform's initial architecture is illustrated on Fig. 3.2.

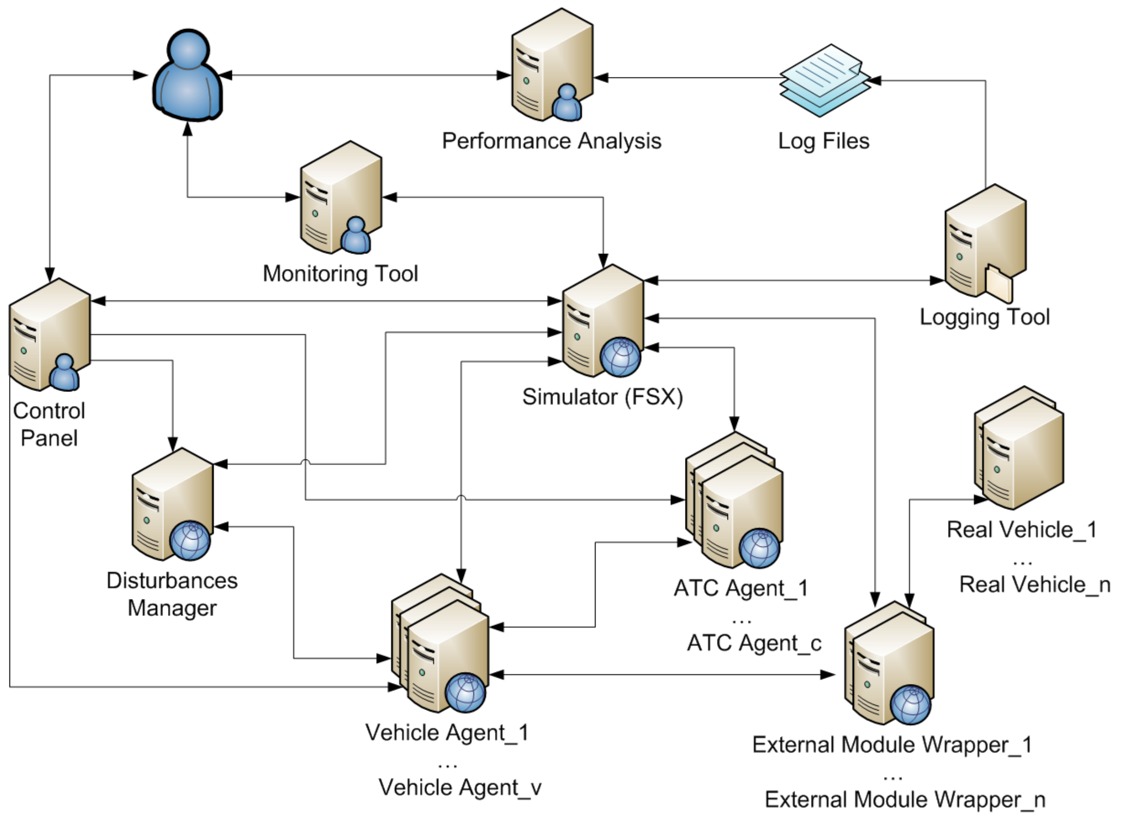


Figure 3.2: The Platform's Initial Architecture

This subsection includes a description of the platform around which the project at hand is centered. It is developed using the C# programming language.

The Platform includes a control panel, which allows for the specification of each mission; an agent to manage traffic (Air Traffic Control, or ATC, Agent); a vehicle control agent; performance analysis; and the basis for a disturbance management tool. The Platform enables the specification and deployment of a wide variety of missions using multiple vehicle types on various simulations. While still technically in prototype stage, the Platform is currently at a fairly advanced level.

The Disturbances Manager is responsible for the simulation of environmental anomalies - which may require the intervention of vehicle teams, or have an impact on unrelated vehicular missions; it communicates with the Platform's Control Panel, vehicle agents, FSX, and optionally with external disturbance-specific simulators (e.g. if one were to use an external tornado simulator as to obtain a more rigorous behavior for such disaster, one could interconnect it with the Disturbances Manager).

At its state by the start of the thesis work, the Control Panel allowed for the basic specification of disturbances using a graphical interface. The disturbances are stored as XML files for the DM to subsequently process; this specification is to be further developed, as to encompass the details of all the disturbances to be considered. Given that the DM is to be fully integrated into a distributed simulation platform, similar architectures [Pereira and Rossetti, 2012][Chen et al., 2006] have been studied and taken into consideration on how to perform said integration.

3.5.2 Programming Languages

Given that **C#** is the focal programming language of The Platform, with the purpose of facilitating the integration of this dissertation's practical work therein, C# will be used to implement all the required DM components.

The DM's graphical interface includes an embedded Google Maps window. Through **Javascript**, the disturbances' affected areas, as well as the vehicles' trajectories, are drawn over the map itself.

3.5.3 AgentService

AgentService [1] is a programing framework for the development of multi-agent systems. The framework is based on the Common Language Interface[2] (CLI) and C#. Provided by the Laboratory of Informatics of the Department of Communication Computer and System Sciences of the University of Genoa, the framework handles the communication between the agents on the Platform. AgentService is FIPA(Foundation for Intelligent Physical Agents)-compliant.

3.5.4 Flight Simulator X

A **Flight Simulator** [Wong, 1998] is a type of virtual reality environment where the user pilots a virtual airplane - usually matching an existing model and simulating its cockpit, dimensions and flight behaviour. They are used for training of prospective pilots and within the context of video-games, with varying degrees of realism.

[1]Available at http://www.agentservice.it/

[2]More details at https://www.iso.org/standard/42927.html

Figure 3.3: Flight Simulator X

Microsoft's Flight Simulator X, originally released in 2006, is a flight simulation video game for Microsoft Windows operating systems. It is the tenth version of the popular "flight sim". The inclusion of a wide array of playable aircraft, all with a detailed, realistic and interactive cockpit; thousands of real-world airports; weather effects; support for other (i.e. land and water) vehicles; compatibility with varied joysticks and other controllers.

FSX is not only a popular video game and simulation platform, but also a good basis for research projects: it provides a solid simulation platform, very complete and well documented API and SDK, capability to simulate specific failures in the vehicles, multiple vehicle types, and more. A *screenshot* of the game's interface can be seen on Fig. 3.3.

Chapter 4

The Disturbance Manager

The present chapter includes a description of The Platform's Architecture and how the DM fits into it, and the functionalities of the Disturbances Manager. Furthermore, the additions to the Disturbance Description language and the Teams Description Language - used to specify details of the team such as the types of vehicles and which sensors they carry - are also expressed.

The current chapter is also used to provide details on the several facets of the DM, specifically on its implementation process. Firstly, a high-level description on the major steps performed by the DM, followed by the technical aspects: the core, the generation of the sensor readings for the vehicles and communication with the remaining components of the platform.

4.1 The Platform's Architecture

This section aims to illustrate the expansion of the initial system architecture by the addition of the components necessary to the successful implementation of the project at hand. Namely the aforementioned optional and external co-simulator, to be used by someone who wishes to use an external disturbances manager as a way to achieve a more accurate representation of the disturbance in question.

In order to facilitate an approach which allows for compatibility with as many as possible possible external, an interface will be developed. The technical details have not been decided, however the HLA architectural specification shows great potential for this type of application - as described in section 2.2. Fig. 4.1 illustrates the expected final architecture for The Platform.

In Fig. 4.1, the External Disturbance Simulator is illustrated, being part of the planned architecture. Several options regarding external simulators were studied in Environmental Disturbance Simulators, however, due to reasons detailed on Final Work Overview, this element was not implemented.

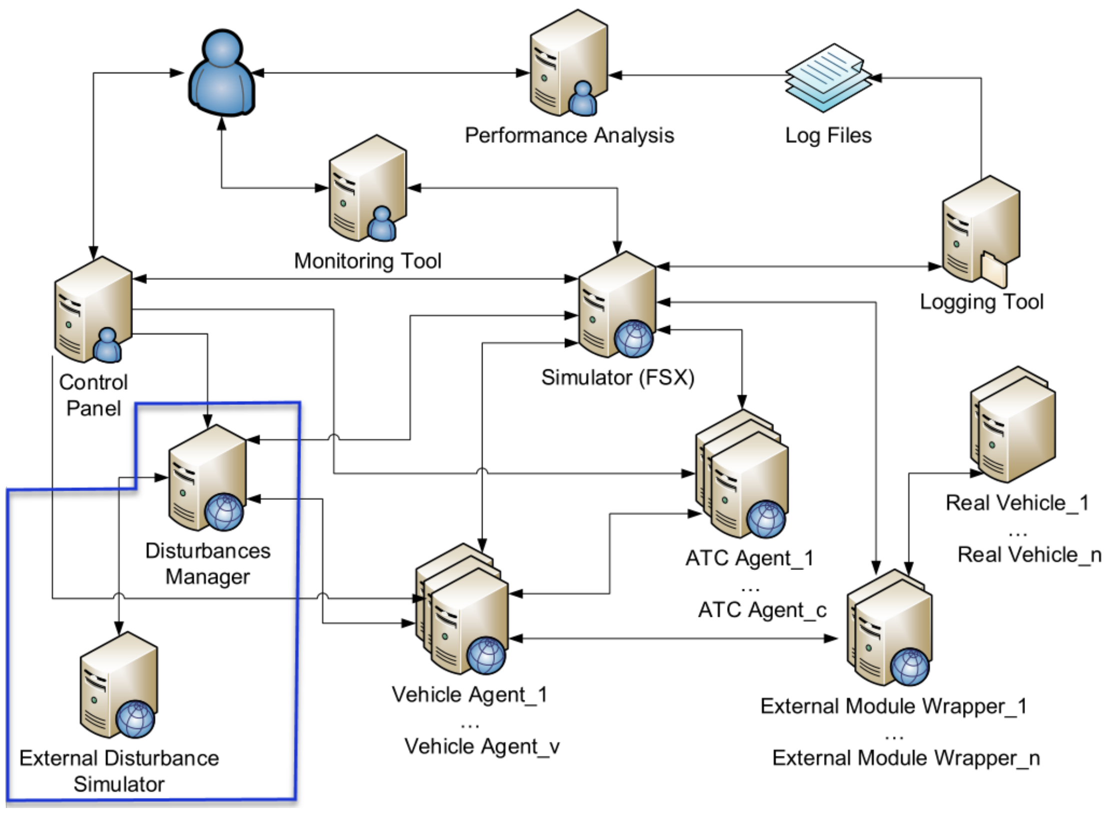


Figure 4.1: The Platform's New Architecture

4.2 The Disturbances Manager

The internal structure of the Disturbances Manager is illustrated in Fig. 4.2. The Platform Communication section (in section 4.9) encompasses the modules responsible for the communication with the other components of The Platform, namely the SimConnect Module (communication with FSX; see subsection 4.9.1) and the AgentService Module (used to communicate with the Vehicle Agents; see subsection 4.9.2). The Core (see section 4.7) components are the *DistrubancesManager* class, which contains the configuration data transferred from the Control Panel (see section 4.9.3) and the *DeployedDisturbance* class, which refers to a Disturbance that is being simulated - containing data not only on the disturbance at hand but also on its evolution.

The main function of the Disturbances Manager is to simulate each disturbance selected by the user of The Platform. The simulation consists on several aspects, including managing the area being affected by the disturbance and communicating the sensor readings to each vehicle (through the corresponding Vehicle Agent); computing the evolution of the disturbance through both time - it may be constant, or evolve as specified by a given function (e.g. exponential, linear, logarithmic) - and space, since it may also increase in size and move through the map. The Disturbances Manager is launched through The Platform's Control Panel, as illustrated in Fig. 4.9, on which the specification of the several aspects for the disturbances are selected by the user, before the actual simulation or deployment occurs. At the time of the instantiating of the DM, it receives input data

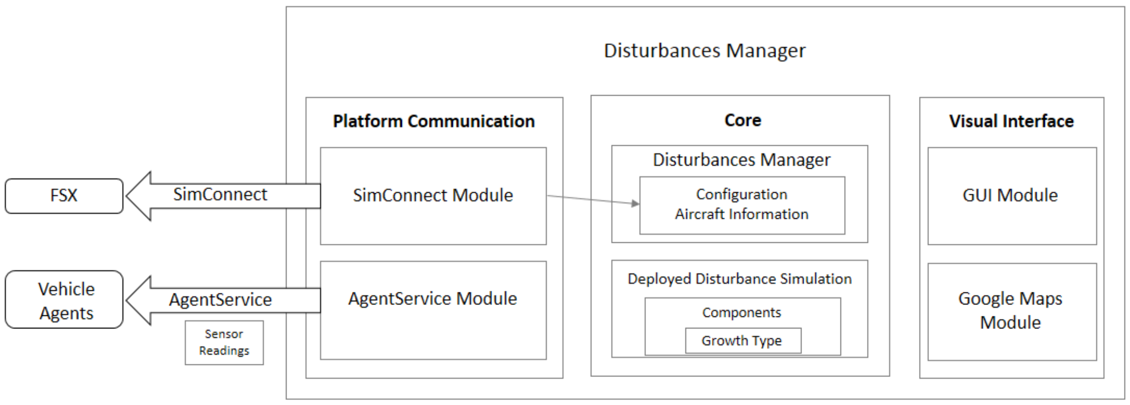


Figure 4.2: The Disturbance Manager's Internal Structure

from the Control Panel - as illustrated in Fig. 4.1 and detailed on subsection 4.9.3.

4.3 Disturbance Description Language (DDL)

The DDL is used to specify disturbances, including their descriptions, areas of effect, type of evolution, and more. An XSD (XML Schema Definition) file formally describes the DDL, which had already been [Silva et al., 2016] prior to the start of the project in progress, however some additions were made as to achieve the planned solution.

The *seriousness* value is an optional integer, ranging from 0 to 5, indicating the degree of severity of the disturbance in question. The *origin* value is a string which carries one of three possible values: *Natural*, *ManMade* and *Any* - relating to the possible origins of the disturbance being specified. For example, an oil spill is an occurrence inherently related to human activities, therefore *ManMade*; while a forest fire may also have (albeit less frequently) a natural origin, therefore *Any*.

The additions to the pre-existing DDL schema are highlighted on Fig. 4.3 (for clarity reasons and size constraints, not all branches of the diagram have been fully expanded) and the following enumeraton includes a short description for each field.

1. *disturbanceTypeGroup/disturbanceType* - This field is used to indicate the type of the disturbance, among those indicated in the tables on 2.1;

2. *distOriginGroup/seriousness* - This field is used to specify the Seriousness Degree of each disturbance, as seen in section 2.1;

3. *distOriginGroup/origin* - As with the previous entry, this correlates to the origin of the disturbances, indicated in Disturbances - Environmental and Beyond;

4. *generatedReading/type* - The type of the generated sensor reading at hand, e.g. a floating point number;

5. *generatedReading/startValue* - The starting value of the emmitted sensor readings;

6. *generatedReading/minLimitValue* - The minimum limit value for the generated readings, even if the growth function returns a lower value;

7. *generatedReading/maxLimitValue* - Analogous to the previous entry, however refers to the maximum limit;

8. *growthFunctionGroup_optional/growthFunction* - along with *functionVariable*, this optional field is used to specify the growth function followed by a specific disturbance component. It is optional because a constant component does not have an associated growth.

An example DDL XML file is included in Appendix A.

Regarding the sensors illustrated on Chapter 2 (Tables 2.1, 2.2, 2.3 and 2.4), the original (i.e. existing before the start of the project at hand) *common.xsd* schema included a few sensors - however more were introduced as to cover all of them. The additions to the XSD file are included in XSD Additions.

4.4 Teams Description Language (TDL)

The TDL indicates information about a vehicle team on the platform: which vehicles it contains, restrictions such as no-fly areas, among other details. An XSD file formally describes the TDL, which - as with the DDL - had already been specified [Silva et al., 2017] prior to the start of the project in progress, however some additions were made as to achieve the planned solution - adding details to the sensors on the team's vehicles.

The fields added to the TDL are the following:

1. *sensorReadingType/type* - Refers to the type of reading generated by the sensor at hand. E.g. a CO_2 sensor will generate *numeric* readings;

2. *sensorReadingType/unit* - Refers to the unit of the generated readings. For example a CO_2 sensor may generate *grams/m*^{3} readings.

These additions to the previously-existing TDL schema are highlighted on Fig. 4.4 (for clarity reasons and size constraints, the remainder of diagram is not shown). An excerpt of an XML TDL file is included in Appendix B.

4.5 Disturbance Database

Besides the tables in chapter 2, it was decided to portray the same information in a more "machine-friendly" manner, therefore two small XML databases were created: one for the disturbances and one for the sensors. The main purpose being the posterior auto-completion of the Control Panel's GUI (Graphical User Interface) - once someone selects a specific disturbance name from the menu, the remainder of the specification (i.e. *origin*, *sensorType*, *seriousnessDegree* and *growthRate*) should be automatically filled with the corresponding values. This way, the user will have the freedom to change the preset values, but won't be affected by the inconvenience of manually setting every aspect of the disturbance to be deployed. The mentioned databases are included in XML Databases.

4.6 High-Level Description

What follows is an enumeration of the main steps needed to use the DM and performed by it. The section is meant to be understandable by someone with no Computer Engineering knowledge, therefore technical terms are purposefully avoided and simpler vocabulary is put to use.

1. The user launches Flight Simulator X and The Platform;
2. The user utilizes the Control Panel of The Platform to define a specific disturbance (for example a fire): where it starts, how large it is and how it grows;
3. The Control Panel checks if all the data was introduced with no errors;
4. The Control Panel saves that information and launches the Disturbances Manager, transmitting to it the information regarding the fire;
5. The Disturbances Manager opens a window which shows the map and all the active vehicles (both on the map and on a table);
6. The Disturbance Manager "informs" the vehicles within the areas affected by the disturbance of the sensor readings they are receiving (Carbon Dioxide and Temperature, for example);
7. The vehicles receive the information and save it on a text file.

4.7 Core

When launched, the DM performs the following key actions:

- A series of initializations, both regarding the GUI elements, the embedded Google Maps window and the connection data required to connect to FSX through SimConnect and to AgentService;
- Loads the disturbances file transferred by the Control Panel and validates it against the corresponding XSD schema;
- Loads the active team configuration file, in order to load the vehicles therein to the GUI and reads each of their sensors in order to later create the column headers on the table identified with *(H)* on Fig. 4.5;

- Dynamically creates the table identified with *(H)* in Fig. 4.5: a column for the vehicle name; another for indicating whether it's active (a vehicle may be on the team but not deployed); the "Affected?" column - which indicates whether a vehicle is under the influence of at least one disturbance; and finally one column for each of the sensors at hand: the ones mentioned in the active team configuration file, as mentioned above, and those listed in the affected sensor types for each of the loaded disturbances' components;

- The disturbances and their components are loaded onto table *(C)* in Fig. 4.5;

- Through SimConnect, an invisible smoke-emitting airplane is spawned in-game at the center of each of the affected areas by the deployed disturbances. The reasoning for this step is justified in the conclusion of chapter 2, and the details of the matching implementation can be read below, on subsection 4.9.1;

- Each time the active vehicle data (table *(I)*) is updated from FSX through SimConnect (details on 4.9.1)

  - Table *(I)* is updated with details on the vehicles active in the simulation (mainly latitude, longitude, altitude, heading and speed);

  - The new vehicles' positions are matched against the areas covered by the disturbances' components (illustrated as the rectangle and the circle in Fig. 4.5-*(F)*): if the vehicles are in the affected area, the *Affected?* value is updated accordingly (Fig. 4.5-*(H)*) and the sensor reading messages are sent to the vehicles (more details in the sections below);

  - The active vehicle's position is updated on the Google Maps window (Fig. 4.5 - *(F)*) by drawing its trajectory as a red line on the map.

4.7.1 GUI

The DM's GUI was built iteratively, with its components being added throughout the development of its features. Illustrated in Fig. 4.5, it is composed of several components listed below.

The DM is launched through the Control Panel; after that, the user should click to connect (if in test mode, otherwise the connection is performed automatically) to FSX through SimConnect and to AgentService. Then, the loaded Disturbances are deployed, either automatically or by pressing the *Deploy* button (test mode).

(A) **AgentService Connection** - Includes text fields which indicate the details for the connection with AgentService. The fields are auto-filled with the data which the Control Panel obtains from The Platform's data on Windows' registry, if available. The connection is performed after the user clicks on the *Connect* button;

(B) **SimConnect Connection** - Utilized for the specification of the IP (Internet Protocol) address of the machine running FSX to which the connection is intended and buttons to connect and disconnect from it. As with the AgentService region, the default IP address is loaded from a Windows registry key;

(C) **Active Disturbance Table** - Indicates the designation of the active disturbances, as well as that of each of their components. Also shows the vehicle sensors affected by the listed components;

(D) ***Center on Disturbances*** **Button** - This button centers the Google Maps window *(F)* on the center of the disturbance being simulated;

(E) ***Deploy*** **Button** - Used only in test mode, activates the disturbance loaded from the Control Panel to the DM, initiating the simulation. While not in test mode, the disturbances run automatically;

(F) **Google Maps Window** - This region of the UI contains an embedded Google Maps page, on which several overlays are created: geometric shapes regarding the affected areas of the disturbances (such as the blue circle and rectangle in Fig. 4.5) and red lines for the trajectory of the vehicles;

(G) **Sensor Readings Message Log** - This region encloses a log of all the sensor readings sent through AgentService throughout the simulation. Each entry includes the identifier for the receiving agent, the message structure, the *sensorid* (identifying the affected sensor) and the *reading* value;

(H) **Team Vehicle Sensor Table** - This table is dynamically generated, which means that instead of having fixed columns with set names, the columns are generated on runtime (the process is described in section 4.7). Includes an identifier for each vehicle in the team, a *Yes* or *No* value on wether it is deployed in the current simulation, and the latest sensor readings for each of the available sensors;

(I) **Active Vehicle Table** - The final table includes the data obtained from FSX through SimConnect (see subsection 4.9.1), namely the vehicle's name, model, ID and several other details, such as position and speed.

4.7.2 Space and Time Evolution of a Disturbance

On the disturbances' XML files, the growth functions are specified along with a *startValue*, a *minLimitValue* and *maxLimitValue*, and a *growthFunction*. In order to demonstrate how the evolution processing is handled, the *linear* type of *growthFunction* with a specific *coefficient* will be used as an example. At each time the update occurs, the following operation is performed for each component of the active disturbances:

$currentValue = startValue + coefficient * (stopwatch.ElapsedMilliseconds/timeScale)$
$if(currentValue < minLimitValue)currentValue = minLimitValue$
$if(currentValue > maxLimitValue)currentValue = maxLimitValue$

The indicated operations allow for the evolution of the disturbance reading emission values through the linear function with the specified coefficient, within the range delimited by the *minLimitValue* and *maxLimitValue* values.

4.8 Vehicle Sensor Readings

The sensor readings, which are sent to the corresponding vehicle agents through AgentService (subsection 4.9.2 for details on the transmission of the message), are calculated through the following formulae:

$$reading = EmmitedReading * (1 - \frac{distanceFromCenterOfDisturbance}{MaxDistanceFromCenterOfDisturbance}) \tag{4.1}$$

$$transmittedReading = \begin{cases} minLimitValue, & \text{if } reading < minLimitValue \\ maxLimitValue, & \text{if } reading > maxLimitValue \end{cases} \tag{4.2}$$

On the first formula, the fraction generates a linear decrease in the *reading* value, as the distance from the center of the disturbance increases. Which means that if such distance is null, then the *reading* value will be the same as *EmmitedReading*; when the *distanceFromCenterOfDisturbance* is at its maximum value, the fraction will equate to 1, as such the *EmmitedReading* value will be null.

Each disturbance's component, on the disturbances XML file, includes a *startValue*, *minLimitValue*, *maxLimitValue*, *growthFunction* and one or more *coefficient*s.

EmmitedReading matches the reading which is emmited at the center of the disturbance, where the vehicles receive the highest readings. When the vehicles get farther from the center, the readings they receive are lower.

DistanceFromCenterOfDisturbance refers to the distance from the vehicle to the center of the disturbance's component which affects it. *MaxDistanceFromCenterOfDisturbance* is the maximum possible distance from a vehicle to the center of the disturbance: if the shape is circular, the value will be the radius; in case of a rectangular region, the value will be half of its diagonal length.

The mentioned reading values are sent to each Vehicle Agent that possesses the corresponding sensor and is contained in the affected area. The Vehicle Agents generate a log, which includes the time when the simulation started, and the received messages in CSV (Comma Separated Values) format.

The interaction regarding the dispatching of the sensor readings is illustrated through the Sequence Diagram in Fig. 4.6.

4.9 Communication with The Platform

The present section includes the details on the communication between the DM and other components of The Platform, namely FSX through SimConnect and the Vehicle Agents through AgentService.

4.9.1 FSX - SimConnect

The DM connects with FSX through SimConnect with two purposes:

- Acquiring information regarding which vehicles are currently deployed;
- Deploying the invisible airplane in the center of the disturbance-affected areas, as to show smoke or other effects which are emitted from the aircraft.

Since other Platform components, such as the ATC Agent, already connected to FSX through SimConnect, the corresponding code was adapted to perform the DM's connection - thereby saving a significant amount of time. A sample project provided in FSX's SDK to obtain details on the position and speed of aircraft also proved useful for the task at hand.

4.9.2 AgentService

The DM connects to the AgentService with the objective of sending sensor readings to the deployed Vehicle Agents. As with SimConnect, the ATC Agent also performs an equivalent connection. Thereby the corresponding code was adapted to perform the DM's connection to the AgentService.

After both connections are complete (SimConnect and AgentService), the obtained information is cross-referenced as to match SimConnect data with AgentService data. This allows for the adding of the AgentService identifier and the SimConnect identifier to the internal data structure used to enclose the vehicle data, such as position and speed.

4.9.3 Control Panel

As the user defines a disturbance through the Disturbance Configuration tab on the Control Panel, or loads one from an XML file to the GUI, and clicks "Launch", the Control Panel does as follows:

1. Saves the Disturbance specified in the GUI, validating it in the process;
2. Attaches the path to the saved Disturbance (XML) file to an Application Configuration (*AppConfig*) structure;
3. Attaches to the same *AppConfig* file the information required for the DM to use the AgentService, i.e. User Name and Password;
4. Launches the DM agent with the complete AppConfig file as a parameter.

4.10 Summary / Conclusions

This chapter detailed the performed development as to achieve a functional Disturbances Manager. A high-level description of the component was presented; followed by the core functions of the DM; the GUI; the processing of space and time evolutions of disturbances; the generation of the sensor readings for the vehicles; the communication with the FSX, AgentService, Control Panel and the Vehicle Agents.

Several aspects were considered throughout the development, from the code to the user interface. These include the usage of relevant comments detailing each function, explicit function and argument names, consistent indentation and appropriate partitioning of code in several files. This should provide future users of The Platform with the means of easily understanding the internal functionality of the DM, as well as the capacity to add features and integrate the DM into future projects.

Overall, the development stage of the Disturbances Manager is considered successful, and was conducted according to the work plan.

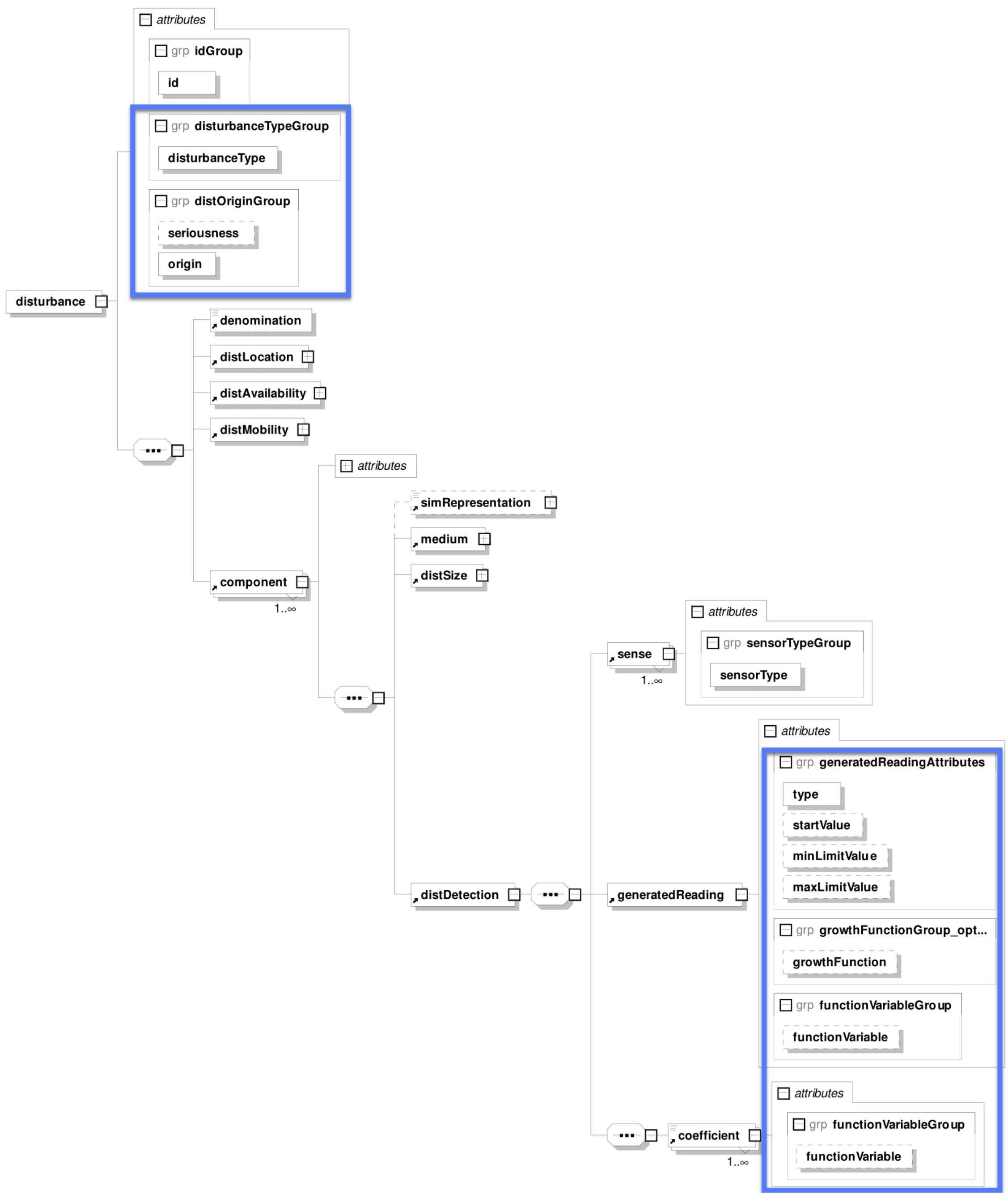


Figure 4.3: DDL Schema Diagram

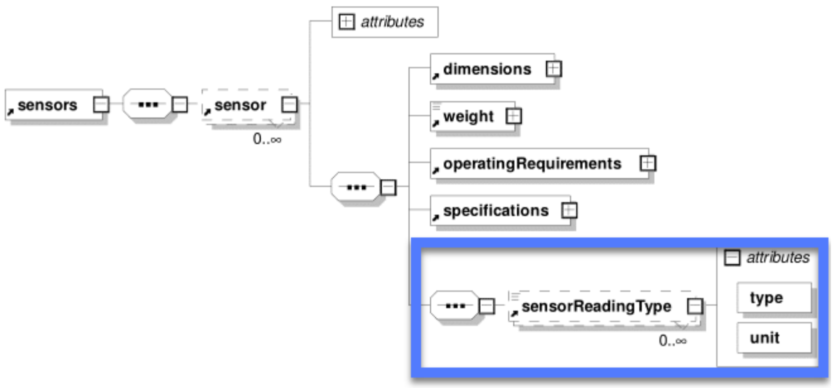
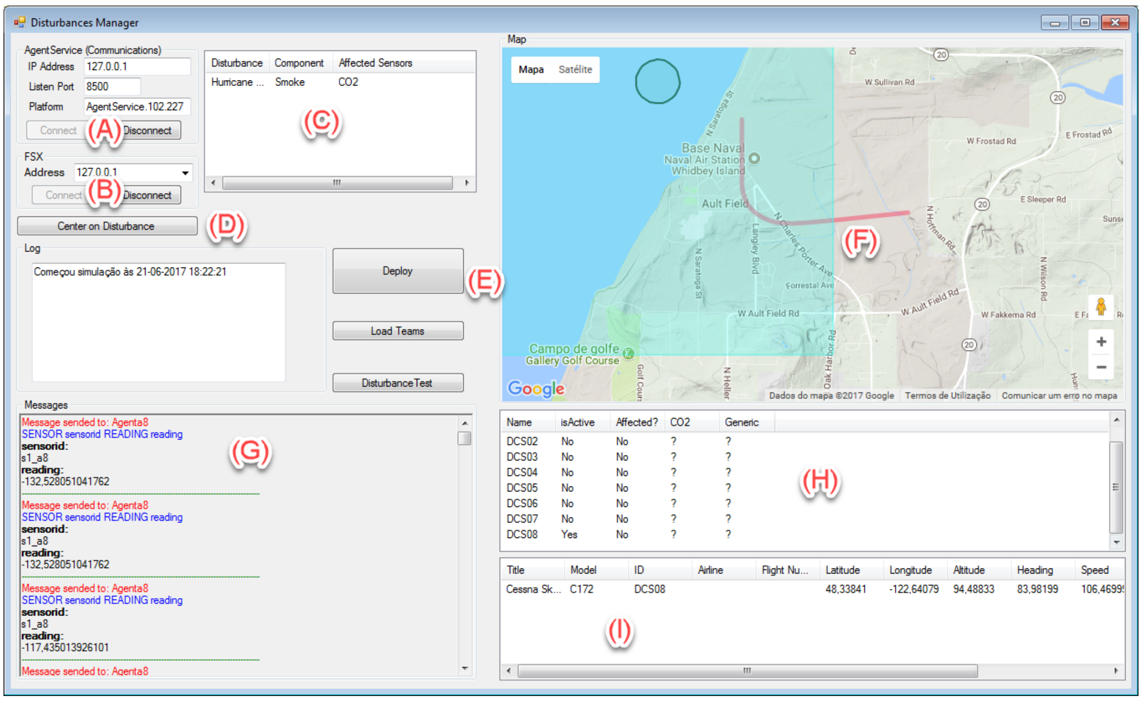


Figure 4.4: TDL Schema Diagram



IP Address	127.0.0.1
Listen Port	8500
Platform	AgentService.102.227
Address	127.0.0.1

Disturbance	Component	Affected Sensors
Hurricane ...	Smoke	CO2

Começou simulação às 21-06-2017 18:22:21

```
Message sended to: Agenta8
SENSOR sensorid READING reading
sensorid:
s1_a8
reading:
-132,528051041762
------------------------------------------
Message sended to: Agenta8
SENSOR sensorid READING reading
sensorid:
s1_a8
reading:
-132,528051041762
------------------------------------------
Message sended to: Agenta8
SENSOR sensorid READING reading
sensorid:
s1_a8
reading:
-117,435013926101
------------------------------------------
Message sended to: Agenta8
```

Name	isActive	Affected?	CO2	Generic
DCS02	No	No	?	?
DCS03	No	No	?	?
DCS04	No	No	?	?
DCS05	No	No	?	?
DCS06	No	No	?	?
DCS07	No	No	?	?
DCS08	Yes	No	?	?

Title	Model	ID	Airline	Flight Nu...	Latitude	Longitude	Altitude	Heading	Speed
Cessna Sk...	C172	DCS08			48,33841	-122,64079	94,48833	83,98199	106,4699

Figure 4.5: The DM's GUI

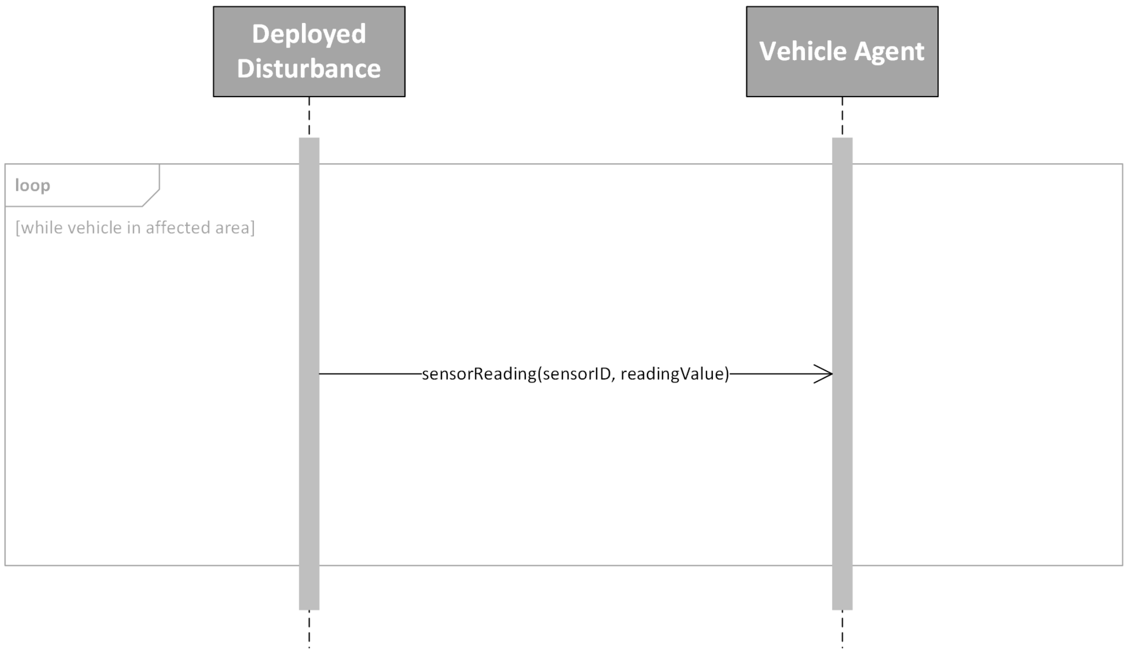

Deployed Disturbance
Vehicle Agent
loop
[while vehicle in affected area]
sensorReading(sensorID, readingValue)

Figure 4.6: Sensor Readings Sequence Diagram

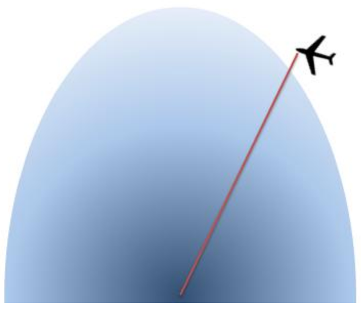


Figure 4.7: Distance to Center of Disturbance (retrieved from [Silva, 2011])

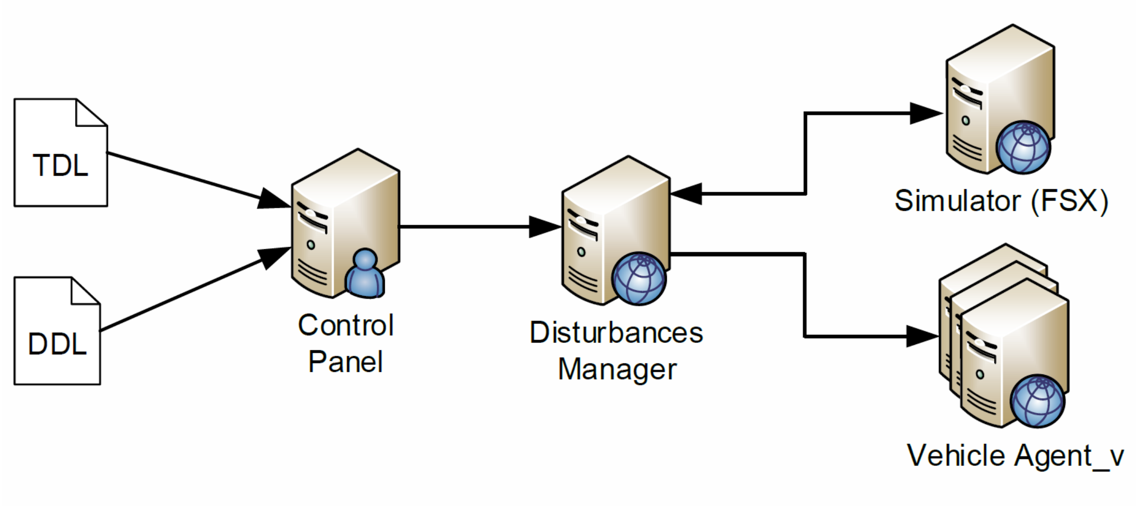
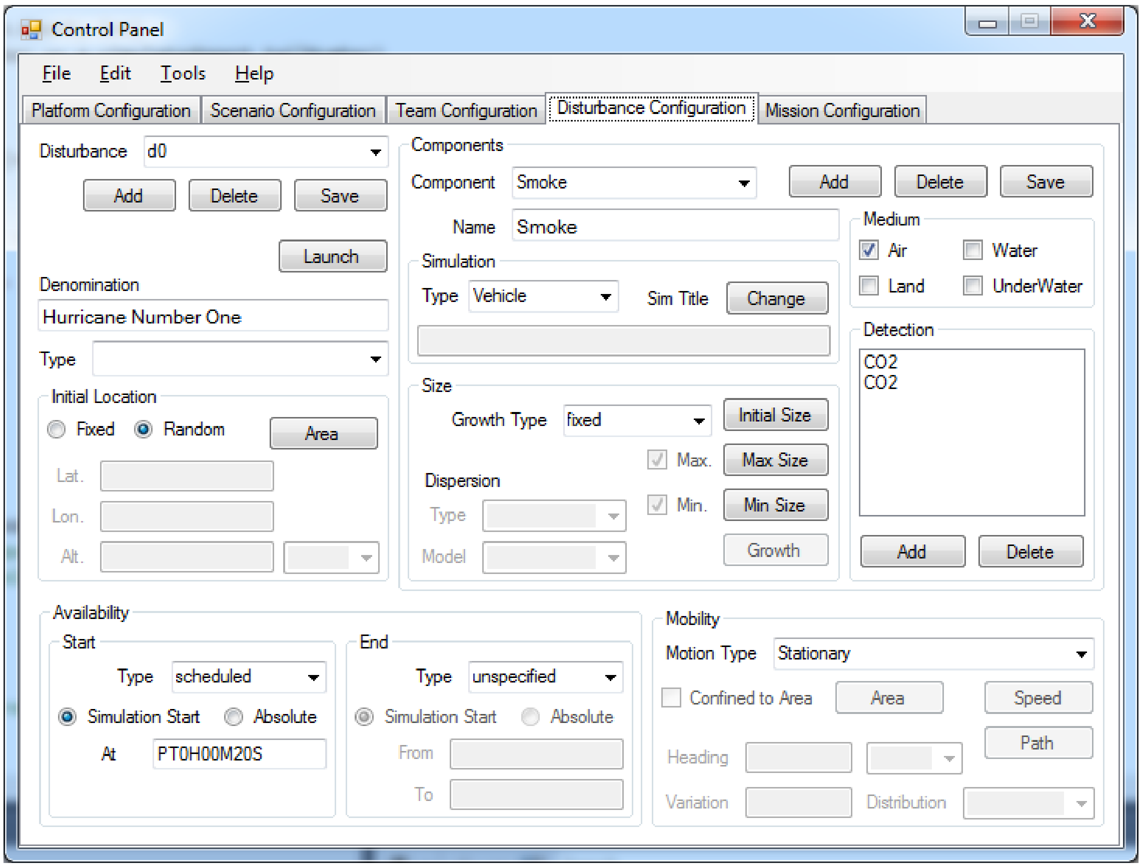


Figure 4.8: The DM's Data Flow (adapted from [Silva, 2011])



Disturbance: d0

Denomination: Hurricane Number One

Type:

Initial Location: Fixed [✓ Random]

Lat.:
Lon.:
Alt.:

Components

Component: Smoke

Name: Smoke

Simulation: Type: Vehicle; Sim Title:

Size: Growth Type: fixed; [✓ Max.] [✓ Min.]

Dispersion: Type: ; Model:

Medium: [✓ Air] Water Land UnderWater

Detection:
CO2
CO2

Availability

Start: Type: scheduled; [✓ Simulation Start] Absolute; At: PT0H00M20S

End: Type: unspecified; Simulation Start Absolute; From: ; To:

Mobility

Motion Type: Stationary

Confined to Area

Heading:
Variation:
Distribution:

Figure 4.9: Control Panel (GUI)

Chapter 5

Testing & Results

The chapter at hand includes information about the creation and performance of the tests developed around the DM and AgentService.

In order to validate the work developed, it is essential to develop a way to test and evaluate its performance. With that objective in mind, the following tests were specified:

1. Core test - the following compose the primary validation for the DM's functionalities. The objective is to verify that all the components are working as intended. Therefore, the following tests will be performed:

   (a) Vehicle Position & Sensor Readings - the most significant test of the DM, meant to verify whether the DM correctly detects if the vehicles are within areas affected by components of disturbances, and if the sensor readings are sent to the corresponding vehicles;

   (b) Higher Readings in The Center - all disturbance components have a center where the intensity for sensor readings is meant to be higher. This test aims to certify if the sensor readings sent to the Vehicle Agents match this aspect;

   (c) Readings Time Evolution - generated sensor readings for the components of disturbances may have an evolution function (linear, polynomial, exponential, etc.). This test's goal is to demonstrate, via matching the sensor readings sent to vehicles, that the mentioned evolution occurs as specified;

   (d) Disturbance Components Space Evolution - the components of disturbances can be static or have specific spacial evolutions. This test aims to corroborate this evolution.

2. AgentService Stress Test - the objective for this test is to find how much of a potential bottleneck AgentService poses to the DM's performance, in terms of messages per agent per second.

3. AgentService Endurance Test - after the conclusion of the AgentService Frequency Stress Test and finding the maximum message rate for AgentService, another test will be run - to find the maximum message rate that AgentService can reliably maintain over an indefinite amount of time.

For all the performed tests, only aircraft vehicles were deployed because these are the focus of The Platform and allow for higher speeds - therefore quicker tests.

5.1 Core Test

5.1.1 Vehicle Position & Sensor Readings

The test at hand, despite being the most significant for the DM, is very straight-forward. In order to validate whether the DM is correctly checking the vehicles' positions, and thereby sending sensor readings to vehicles within affected regions, the following steps will be performed:

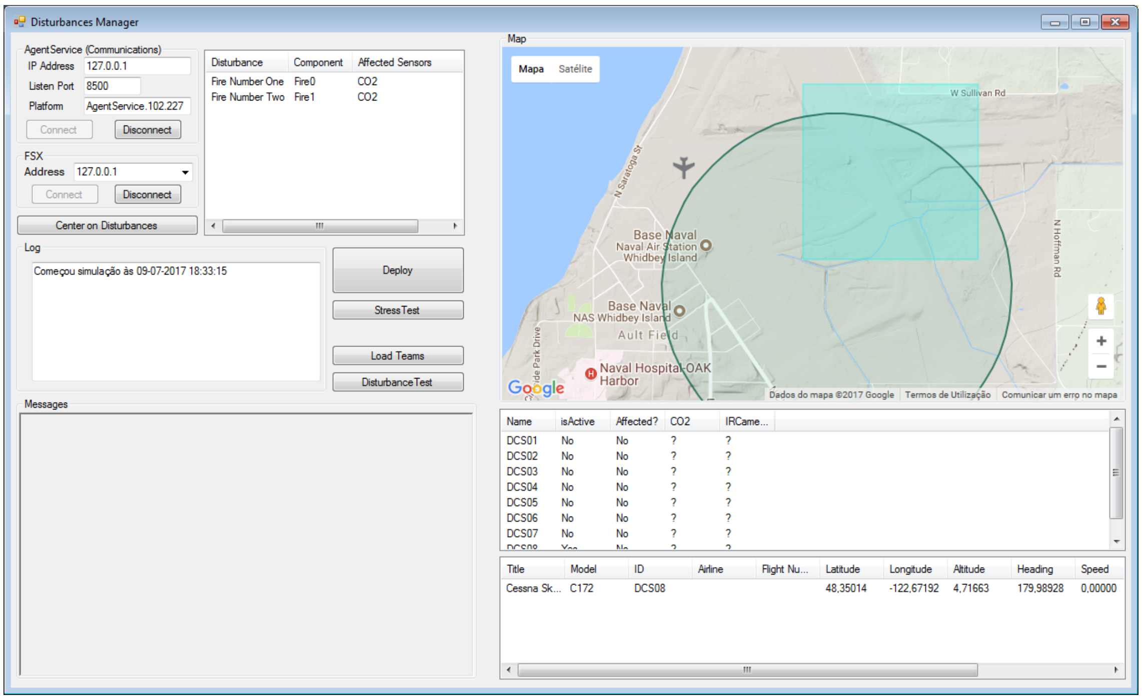
1. A randomly-placed disturbance is deployed;
2. The vehicle is initially placed outside the area affected by a specific disturbance;
3. An instruction is given to the vehicle, in order for it to move toward the affected area;
4. Both visually (through the Google Maps window on the DM's GUI) and manually (through the debug information on the console, checking the coordinates of the vehicle), the point after which the vehicle should start to receive sensor readings can be verified;
5. If the vehicle starts to receive and register sensor readings on the mentioned point and stops doing so after leaving the affected area, the test results are deemed positive.

Confirming what was expected, the vehicle started receiving sensor readings as soon as it entered the affected area visible on the map. On Fig. 5.1, the airplane is outside the visible affected regions (green and blue), and the sensor reading log (bottom left area of the image, see GUI) is empty. On Fig. 5.2 the airplane just entered one of the regions, and began receiving the matching sensor readings, as expected. The vehicle's coordinates were also manually checked, in order to verify whether its coordinates at the time it started receiving messages match those of the outer region of the disturbance.

5.1.2 Higher Readings in The Center

In order to verify whether the generated sensor readings are higher in the center of the disturbance components, the listed steps will be executed:

1. A circular component is deployed on the map;
2. The vehicle is initially placed outside the area affected by a specific disturbance;



AgentService (Communications)
IP Address: 127.0.0.1
Listen Port: 8500
Platform: AgentService.102.227

FSX
Address: 127.0.0.1

Disturbance	Component	Affected Sensors
Fire Number One	Fire0	CO2
Fire Number Two	Fire1	CO2

Log

```
Começou simulação às 09-07-2017 18:33:15
```

Name	isActive	Affected?	CO2	IRCame...
DCS01	No	No	?	?
DCS02	No	No	?	?
DCS03	No	No	?	?
DCS04	No	No	?	?
DCS05	No	No	?	?
DCS06	No	No	?	?
DCS07	No	No	?	?
DCS08	Yes	No	?	?

Title	Model	ID	Airline	Flight Nu...	Latitude	Longitude	Altitude	Heading	Speed
Cessna Sk...	C172	DCS08			48,35014	-122,67192	4,71663	179,98928	0,00000

Figure 5.1: Vehicle Position & Sensor Readings Test - (1/2)

3. An instruction is given to the vehicle, in order for it to move through the center of the affected area;

4. The vehicle should receive readings while in the affected region, and perform the corresponding text logs;

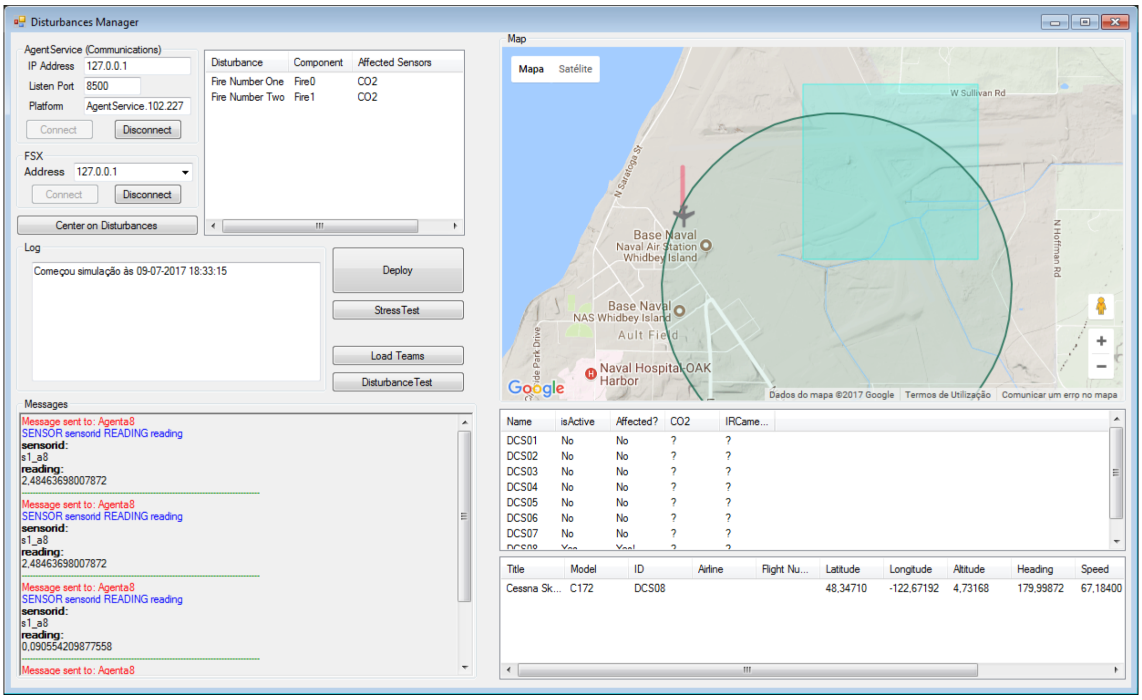
5. After the vehicle leaves the affected region, the text logs are analyzed as to verify that the obtained reading values for the sensor in question grow linearly up to a specific value, and then symmetrically decrease until the vehicle stopped receiving readings. If the vehicle logs match the expected values, this test is validated.

As expected, while outside the affected area (Fig. 5.3), the vehicle did not receive any readings (vertical axis on the figure). Entering the disturbance's region, the sensor readings started to be transferred to the vehicle via AgentService - having the lowest readings at the outskirts of the region in question, and higher its center. The vehicle logs its received readings in CSV format, which enabled straightforward creation of the graph on Fig. 5.3 - which confirms the expected results.

5.1.3 Readings Time Evolution

As to check whether the disturbances' sensor readings can evolve as specified, the aftermentioned steps are to be performed:

1. A disturbance with linear evolution and a specific coefficient is deployed;



AgentService (Communications)
IP Address 127.0.0.1
Listen Port 8500
Platform AgentService.102.227

FSX
Address 127.0.0.1

Disturbance	Component	Affected Sensors
Fire Number One	Fire0	CO2
Fire Number Two	Fire1	CO2

Log

```
Começou simulação às 09-07-2017 18:33:15
```

Messages

```
Message sent to: Agenta8
SENSOR sensorid READING reading
sensorid:
s1_a8
reading:
2,48463698007872
------------------------------------------------------------
Message sent to: Agenta8
SENSOR sensorid READING reading
sensorid:
s1_a8
reading:
2,48463698007872
------------------------------------------------------------
Message sent to: Agenta8
SENSOR sensorid READING reading
sensorid:
s1_a8
reading:
0,090554209877558
------------------------------------------------------------
Message sent to: Agenta8
```

Name	isActive	Affected?	CO2	IRCame...
DCS01	No	No	?	?
DCS02	No	No	?	?
DCS03	No	No	?	?
DCS04	No	No	?	?
DCS05	No	No	?	?
DCS06	No	No	?	?
DCS07	No	No	?	?
DCS08	Yes	Yes!	?	?

Title	Model	ID	Airline	Flight Nu...	Latitude	Longitude	Altitude	Heading	Speed
Cessna Sk...	C172	DCS08			48,34710	-122,67192	4,73168	179,99872	67,18400

Figure 5.2: Vehicle Position & Sensor Readings Test - (2/2)

2. A vehicle is placed within the affected region;
3. The disturbance's generated readings should evolve according to a linear function with the specified coefficient and be saved accordingly by the vehicle;
4. The simulation is terminated and the vehicle logs analyzed. If the logged sensor reading values match the previously specified function, the test's validity is confirmed.

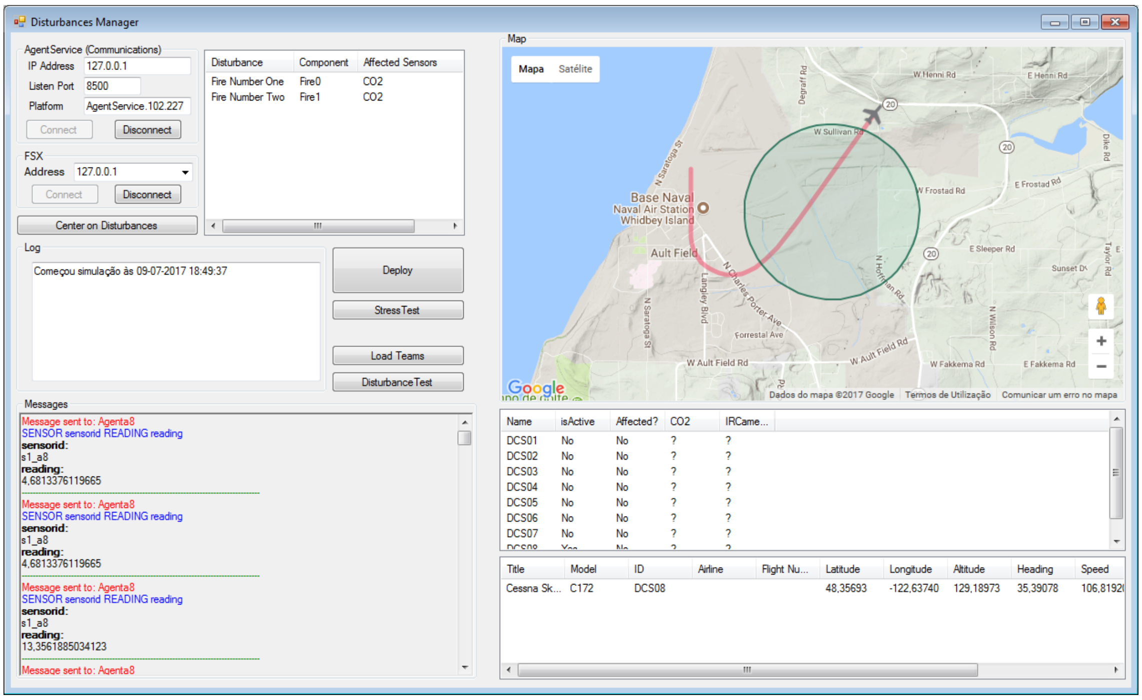
As to perform the test, a stopped airplane was placed within the rectangular area of effect of a disturbance, as displayed in Fig. 5.5. The evolution of the readings for that disturbance was specified as linear, and should have a maximum value of 100.

The received sensor readings of the airplane were converted into a graph and can be observed in Fig. 5.6. As expected, the sensor readings received by the vehicle used in the test matched the specified evolution function. The last three values of the graph match the maximum value of 100 reached by the disturbance; however, since the readings are lower as the vehicles increase their distance to the center of the disturbances, the original value of 100 decreased to around 73.

5.1.4 Disturbance Components Space Evolution

As a means to confirm the spacial evolution of disturbance components, the following steps will be performed:

1. A circular and growing disturbance is specified and deployed;
2. A static vehicle is placed outside the affected region;



IP Address 127.0.0.1
Listen Port 8500
Platform AgentService.102.227

Address 127.0.0.1

Disturbance	Component	Affected Sensors
Fire Number One	Fire0	CO2
Fire Number Two	Fire1	CO2

```
Começou simulação às 09-07-2017 18:49:37
```

```
Message sent to: Agenta8
SENSOR sensorid READING reading
sensorid:
s1_a8
reading:
4,6813376119665
---------------------------------------------
Message sent to: Agenta8
SENSOR sensorid READING reading
sensorid:
s1_a8
reading:
4,6813376119665
---------------------------------------------
Message sent to: Agenta8
SENSOR sensorid READING reading
sensorid:
s1_a8
reading:
13,3561885034123
---------------------------------------------
Message sent to: Agenta8
```

Name	isActive	Affected?	CO2	IRCame...
DCS01	No	No	?	?
DCS02	No	No	?	?
DCS03	No	No	?	?
DCS04	No	No	?	?
DCS05	No	No	?	?
DCS06	No	No	?	?
DCS07	No	No	?	?
DCS08	Yes	No	?	?

Title	Model	ID	Airline	Flight Nu...	Latitude	Longitude	Altitude	Heading	Speed
Cessna Sk...	C172	DCS08			48,35693	-122,63740	129,18973	35,39078	106,8192

Figure 5.3: Higher Readings in The Center Test - (1/2)

3. The disturbance's component's growth is observed through the Google Maps window on the DM's GUI, along with the vehicle's (fixed) position;

4. At the moment the circular component region reaches the vehicle, it should start receiving sensor reading messages - this is to be verified by noticing if this starts to occur at the expected time. If it does, and if the received readings increase according to the formulas on section 4.8 the test is considered covered.

Initially, the disturbance was deployed nearby the placed airplane, as can be observed in Fig. 5.7. As expected, when the circular effect region grew past the airplane's location, the readings started to be transmitted (observable in Fig. 5.8) and increasing, since the airplane's relative distance to the disturbance's center decreases.

5.2 AgentService Stress Test

The XML specification of the vehicle teams include specific definitions of the sensors for each vehicle (as can be seen on the used Team Configuration file on B). This includes several aspects, such as the working temperature and humidity ranges of the sensors, as well as their mass. Another setting that would be an interesting addition to the sensors would be the frequency with which they obtain readings. At the current stage of the DM, the sensor readings are sent to the matching vehicles each time their positions are updated - i.e. at each time the SimConnect information is updated. The current SimConnect refresh rate is around two times per second. Considering the real-life characteristics of sensors, which can vary in terms of refresh rate (slower sensors will

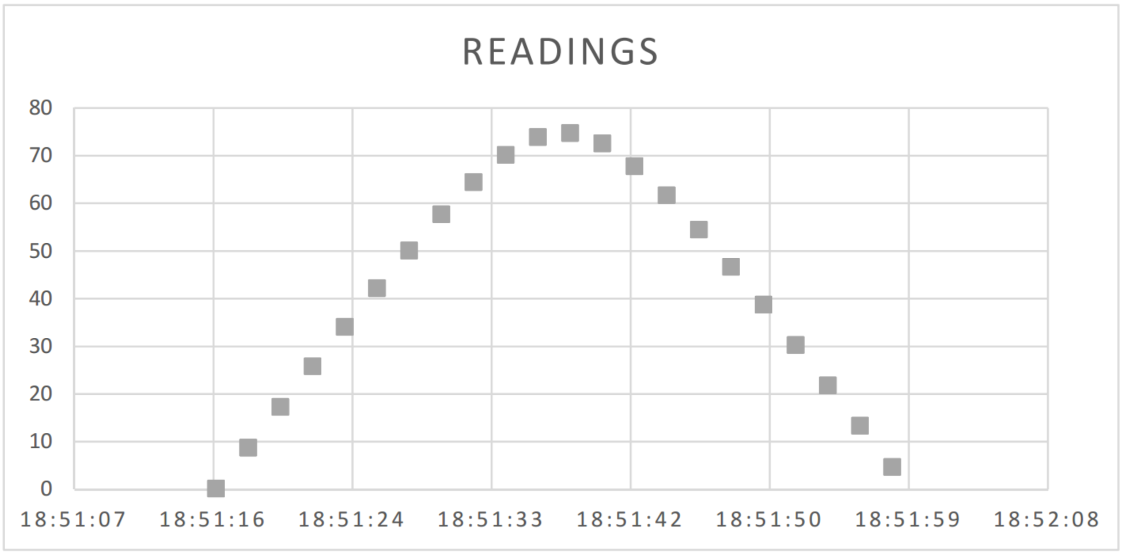


Figure 5.4: Higher Readings in The Center Test - (2/2)

not obtain readings as quickly as a faster sensor), it becomes relevant to take into consideration different refresh rates.

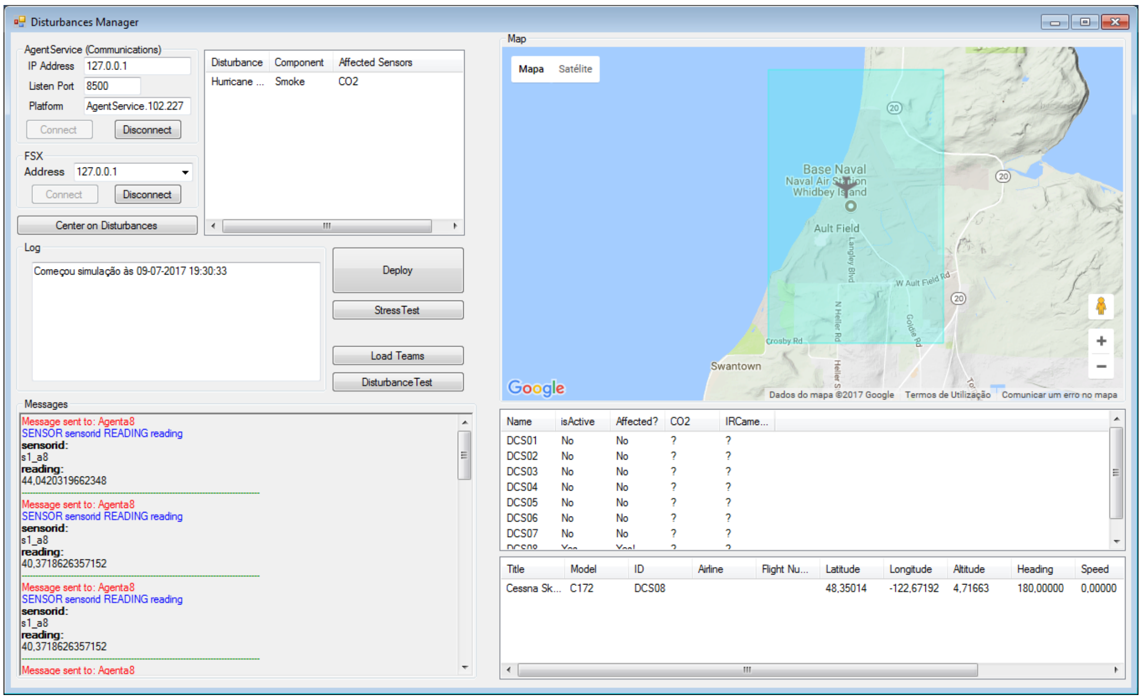
AgentService is software responsible for the transferring of messages from the DM to the Vehicle Agents (see The Disturbance Manager's Internal Structure). As such, it represents a possible bottleneck in terms of the message frequency. Therefore it was considered relevant to perform tests in order to perceive what were AgentService's limits: how many reading messages can be sent per second, and to how many agents. As such, a small script was created in order to simulate specific message frequencies, and find the limit. It is not only relevant to find the limit up to which AgentService is able to run without errors, but also to find the message frequencies that cause delays in the performance of The Platform - i.e. if the messages, meant to be processed in real-time (simulation time) are received without significant delays. However interesting and useful to know how many messages per second AgentService can handle in short bursts, the end-goal of the following tests is to discover a message frequency, for specific numbers of active Vehicle Agents, which can be processed indefinitely in an efficient manner.

The test script is run through the DM, after it connects to FSX and AgentService. The required number of Vehicle Agents are launched through the Control Panel. The simulation variables of the tests are as follows:

- *nMinutes* - the duration of the tests, in minutes
- *messagesPerSecond* - the message frequency, in messages per second

The resulting elements of the tests are the following:

- *Effective Number of Messages/second/agent* - corresponds to the number of messages being sent per second for each agent. It can differ from the *messagesPerSecond* value as, with high



AgentService (Communications)
IP Address 127.0.0.1
Listen Port 8500
Platform AgentService.102.227

FSX
Address 127.0.0.1

Disturbance	Component	Affected Sensors
Hurricane ...	Smoke	CO2

Log
```
Começou simulação às 09-07-2017 19:30:33
```

Messages
```
Message sent to: Agenta8
SENSOR sensorid READING reading
sensorid:
s1_a8
reading:
44,0420319662348
--------------------------------------------------------
Message sent to: Agenta8
SENSOR sensorid READING reading
sensorid:
s1_a8
reading:
40,3718626357152
--------------------------------------------------------
Message sent to: Agenta8
SENSOR sensorid READING reading
sensorid:
s1_a8
reading:
40,3718626357152
--------------------------------------------------------
Message sent to: Agenta8
```

Name	isActive	Affected?	CO2	IRCame...
DCS01	No	No	?	?
DCS02	No	No	?	?
DCS03	No	No	?	?
DCS04	No	No	?	?
DCS05	No	No	?	?
DCS06	No	No	?	?
DCS07	No	No	?	?
DCS08	Yes	Yes!	?	?

Title	Model	ID	Airline	Flight Nu...	Latitude	Longitude	Altitude	Heading	Speed
Cessna Sk...	C172	DCS08			48,35014	-122,67192	4,71663	180,00000	0,00000

Figure 5.5: Readings Time Evolution Test - (1/2)

values, AgentService will originate delays in their transferring; therefore, the *messagesPerSecond* value is the desired message frequency, while the Effective Messages/second/agent value represents the real frequency obtained in the tests, which include AgentService's delays;

- *Errors* - Some tests generated AgentService related errors, and were only successful after a certain number of runs. The number of failed test runs before the achieved results for a specific test is represented in Errors.

The developed test script is detailed in the following steps:

1. The time to wait between sending each message is calculated through the following self-explanatory formula: $secondsBetweenMessages = 1/messagesPerSecond$
2. A stopwatch is started
3. A random value for the simulated sensor reading is calculated
4. A sensor reading message is sent to each of the active Vehicle Agents, containing the randomly generated reading value
5. Waiting for the amount of time calculated in step 1

The script then repeats the loop of generating a random reading value, sending the message and waiting for the number of iterations calculated through the following formula, which equates to the total number of messages sent to each agent:

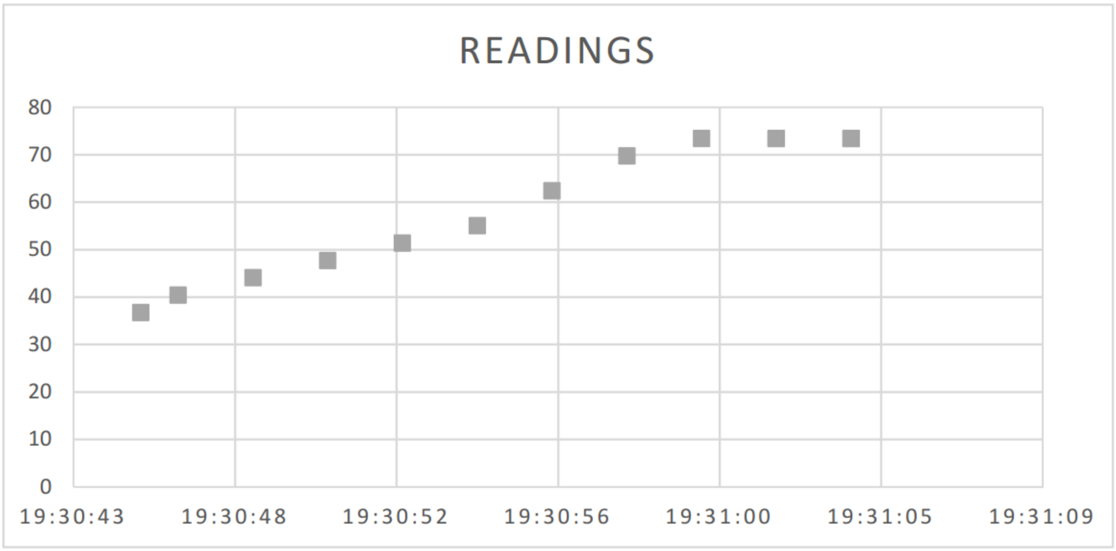


Figure 5.6: Readings Time Evolution Test - (2/2)

$nIterations = nMinutes * 60 * messagesPerSecond$

Multiple test scenarios were performed, and the corresponding results are displayed in Tables 5.1, 5.2 and 5.3, where the scenarios which did not achieve a successful test run after 5+ tries were marked with *ERROR*.

Analyzing the obtained results, Figs. 5.9 and 5.10 were generated, and it became apparent that the test time was largely exceeding the expected time, especially with high message frequencies combined with a high number of Vehicle Agents, e.g. in the Five Minute Stress Test (which should, as the name implies, last five minutes) with 8 agents and 2 messages per second, the elapsed test time was over five minutes. This situation represents a very significant delay, as such the present test method should be optimized.

After some consideration, it was decided to change the way the *secondsBetweenMessages* value was calculated. The current formula, calculating the inverse of the desired message frequency, is only applicable if each message is handled instantly. Since this is not the case, the effective delay will be the sum of the *secondsBetweenMessages* value and the delay inherent to AgentService's processing. As the latter can not be changed, the *secondsBetweenMessages* value obviously required some adapting. As such, the following optimization was performed to the test script, every two seconds:

1. Calculate the effective rate of messages being transferred;
2. If the value is lower than *messagesPerSecond*, the *secondsBetweenMessages* delay is reduced
3. Otherwise, if the value is higher, the *secondsBetweenMessages* delay is increased

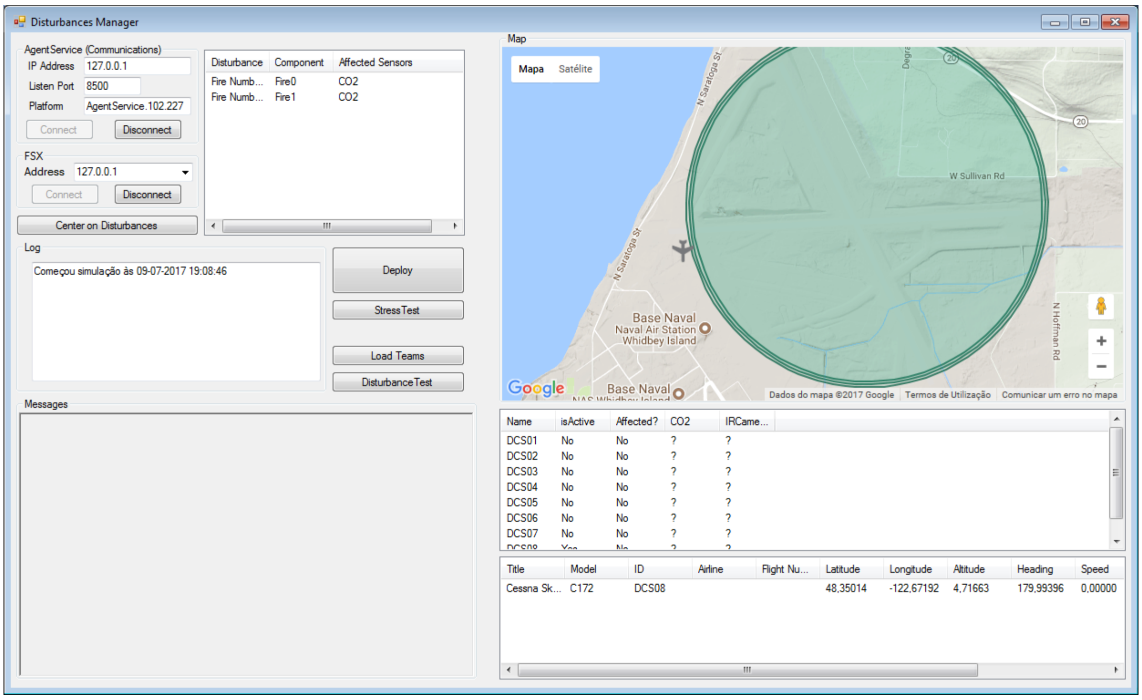
-

Table 5.1: AgentService One Minute Stress Tests

nAgents	nMessages	Messages/second	Errors	Time (minutes)	Effective Messages/second /agent	Effective messages/second
1	600	10	0	1,003	9,974	9,974
1	1200	20	0	1,012	19,759	19,759
1	2400	40	1	1,157	34,569	34,569
1	3600	60	1	1,208	49,673	49,673
1	4800	80	1	1,759	45,493	45,493
1	6000	100	0	1,410	70,933	70,933
1	7200	120	0	2,098	57,193	57,193
1	8400	140	2	1,529	91,540	91,540
1	9600	160	3	1,630	98,150	98,150
1	10800	180	1	1,854	97,067	97,067
1	12000	200	2	1,875	106,662	106,662
1	13200	220	1	2,184	100,750	100,750
1	14400	240	1	2,437	98,493	98,493
1	15600	160	1	2,457	105,819	105,819
1	16800	280	2	2,669	104,901	104,901
1	18000	300	2	2,930	102,398	102,398
1	19200	320	0	3,247	98,564	98,564
2	6000	100	0	1,096	91,228	182,455
2	7200	120	1	1,305	91,935	183,869
4	4800	80	0	2,859	27,981	111,923
4	6000	100	1	2,766	36,152	144,608
8	4200	70	0	4,372	16,009	128,073
8	4800	80	1	5,683	14,077	112,620

Table 5.2: AgentService Five Minute Stress Tests

nAgents	nMessages	Messages/second	Errors	Time (minutes)	Effective Messages/second /agent	Effective messages/second
1	3000	10	0	5,208	9,600	9,600
1	4500	15	0	5,253	14,277	14,277
1	4800	16	0	5,461	14,649	14,649
1	5100	17	1	ERROR	ERROR	ERROR
2	3000	5	0	5,201	9,613	19,226
2	6000	10	ERROR	ERROR	ERROR	ERROR
4	1500	5	1	5,370	4,656	18,622
4	3000	10	0	5,942	8,415	33,660
8	600	2	0	5,477	1,826	14,607
8	900	3	1	5,767	2,601	20,809
8	1500	5	ERROR	ERROR	ERROR	ERROR
8	3000	10	ERROR	ERROR	ERROR	ERROR



AgentService (Communications)
IP Address: 127.0.0.1
Listen Port: 8500
Platform: AgentService.102.227

FSX
Address: 127.0.0.1

Disturbance	Component	Affected Sensors
Fire Numb...	Fire0	CO2
Fire Numb...	Fire1	CO2

Log
Começou simulação às 09-07-2017 19:08:46

Messages

Name	isActive	Affected?	CO2	IRCame...
DCS01	No	No	?	?
DCS02	No	No	?	?
DCS03	No	No	?	?
DCS04	No	No	?	?
DCS05	No	No	?	?
DCS06	No	No	?	?
DCS07	No	No	?	?
DCS08	Yes	No	?	?

Title	Model	ID	Airline	Flight Nu...	Latitude	Longitude	Altitude	Heading	Speed
Cessna Sk...	C172	DCS08			48,35014	-122,67192	4,71663	179,99396	0,00000

Figure 5.7: Disturbance Components Space Evolution Test - (1/2)

This optimization allows for the dynamic adaptation of the delay introduced between messages. Since AgentService's delays increase with the number of agents and message frequency, the updated test script will adapt and assumingly provide better results. It should be noted that, once the *secondsBetweenMessages* delay is reduced as much as it can (zero seconds), it means that the overhead originated through AgentService is fully responsible for the occurring delays.

The optimized test process was run for each test of the previous test scenarios, in order to compare results (listed in tables 5.4, 5.5 and 5.6).

The obtained results were highly improved, as can be seen in Tables 5.4, 5.5 and 5.6 and Figs. 5.11 and 5.12. The difference between the expected and obtained message frequencies (Fig. 5.11) decreased significantly, generally becoming much closer to the desired null value - which occurred

Table 5.3: AgentService Ten Minute Stress Tests

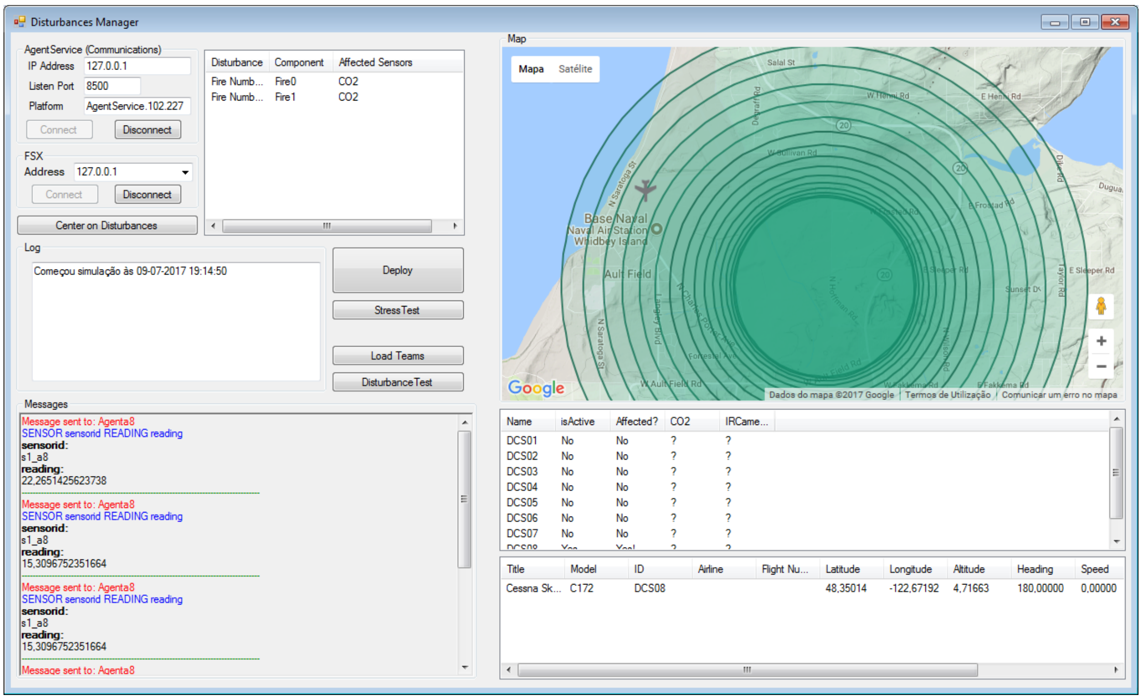
nAgents	nMessages	Messages/second	Errors	Time (minutes)	Effective Messages/second /agent	Effective messages/second
1	6000	10	0	10,0932183	9,90764252	9,908
1	12000	20	ERROR	ERROR	ERROR	ERROR
2	6000	10	0	10,49796551	9,525655095	19,051
2	12000	20	ERROR	ERROR	ERROR	ERROR
4	6000	10	0	12,05686293	8,294031365	33,176
4	12000	20	ERROR	ERROR	ERROR	ERROR
8	6000	10	0	14,99001083	6,671109195	53,369
8	12000	20	0	19,85166051	10,07472391	80,598
8	18000	30	ERROR	ERROR	ERROR	ERROR

Table 5.4: AgentService One Minute Stress Tests (Optimized)

nAgents	nMessages	Messages/second	Errors	Time (minutes)	Effective Messages/second /agent	Effective messages/second
1	600	10	0	0,965	10,368	10,368
1	1200	20	0	1,002	19,969	19,969
1	2400	40	0	1,014	39,447	39,447
1	3600	60	0	1,004	59,749	59,749
1	4800	80	0	1,008	79,388	79,388
1	6000	100	0	1,021	97,977	97,977
1	7200	120	0	1,037	115,703	115,703
1	8400	140	5	ERROR	ERROR	ERROR
1	9600	160	5	ERROR	ERROR	ERROR
1	10800	180	5	ERROR	ERROR	ERROR
1	12000	200	5	ERROR	ERROR	ERROR
1	13200	220	5	ERROR	ERROR	ERROR
1	14400	240	5	ERROR	ERROR	ERROR
1	15600	160	5	ERROR	ERROR	ERROR
1	16800	280	5	ERROR	ERROR	ERROR
1	18000	300	5	ERROR	ERROR	ERROR
1	19200	320	5	ERROR	ERROR	ERROR
2	6000	100	1	1,059	94,435	188,871
2	7200	120	0	1,170	102,576	205,152
4	4800	80	0	1,441	55,506	222,023
4	6000	100	0	1,768	56,577	226,307
8	4200	70	0	5,400	12,964	103,713
8	4800	80	0	5,822	13,741	109,929

Table 5.5: AgentService Five Minute Stress Tests (Optimized)

nAgents	nMessages	Messages/second	Errors	Time (minutes)	Effective Messages/second /agent	Effective messages/second
1	3000	10	0	4,901	10,201	10,201
1	4500	15	0	4,902	15,300	15,300
1	4800	16	2	5,176	15,455	15,455
1	5100	17	1	4,746	17,911	17,911
2	3000	5	1	4,961	5,039	10,079
2	6000	10	0	4,933	10,135	20,270
4	1500	5	0	4,947	5,054	20,215
4	3000	10	0	4,971	10,058	40,233
8	600	2	0	4,943	2,023	16,185
8	900	3	0	4,965	3,021	24,170
8	1500	5	0	4,967	5,033	40,266
8	3000	10	0	4,998	10,003	80,025



IP Address 127.0.0.1
Listen Port 8500
Platform AgentService.102.227

FSX Address 127.0.0.1

Disturbance	Component	Affected Sensors
Fire Numb...	Fire0	CO2
Fire Numb...	Fire1	CO2

```
Começou simulação às 09-07-2017 19:14:50
```

```
Message sent to: Agenta8
SENSOR sensorid READING reading
sensorid:
s1_a8
reading:
22,2651425623738
------------------------------------------
Message sent to: Agenta8
SENSOR sensorid READING reading
sensorid:
s1_a8
reading:
15,3096752351664
------------------------------------------
Message sent to: Agenta8
SENSOR sensorid READING reading
sensorid:
s1_a8
reading:
15,3096752351664
------------------------------------------
Message sent to: Agenta8
```

Name	isActive	Affected?	CO2	IRCame...
DCS01	No	No	?	?
DCS02	No	No	?	?
DCS03	No	No	?	?
DCS04	No	No	?	?
DCS05	No	No	?	?
DCS06	No	No	?	?
DCS07	No	No	?	?
DCS08	Yes	Yes!	?	?

Title	Model	ID	Airline	Flight Nu...	Latitude	Longitude	Altitude	Heading	Speed
Cessna Sk...	C172	DCS08			48,35014	-122,67192	4,71663	180,00000	0,00000

Figure 5.8: Disturbance Components Space Evolution Test - (2/2)

for lower message frequencies, as expected on account of the smaller overhead due to a lower rate of messages.

The information provided thus far represents the conclusion of AgentService's stress testing. The results were not as good as expected, since many errors occurred when trying to raise the rate of the messages. Another negative aspect was the inconsistent occurrence of errors, since many tests only performed successfully after a few attempts.

Table 5.6: AgentService Ten Minute Tests (Optimized)

nAgents	nMessages	Messages/second	Errors	Time (minutes)	Effective Messages/second /agent	Effective messages/second
1	6000	10	0	9,700	10,309	10,309
1	12000	20	0	9,412	21,250	21,250
2	6000	10	0	9,822	10,181	20,361
2	12000	20	1	9,736	20,541	41,083
4	6000	10	0	9,921	10,079	40,317
4	12000	20	0	9,814	9,814	39,255
8	6000	10	0	10,074	9,926	79,411
8	12000	20	0	12,869	15,541	124,332
8	18000	30	0	18,134	16,544	132,350

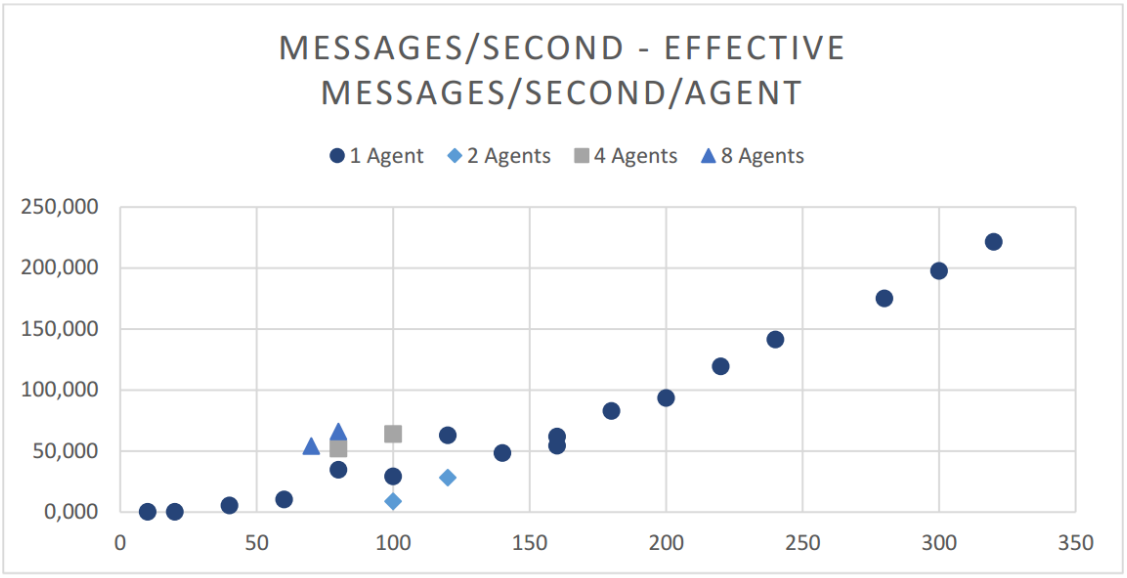


Figure 5.9: Difference Between Expected Message Rate and Effective Message Rate

5.3 AgentService Endurance Tests

For any number of agents (one to eight being the most typical uses of The Platform), the maximum message rate that enables error-free performance of the DM in The Platform would be defined as one which could be used without errors, at the first try and for an indefinite amount of time. As so, an additional set of tests were performed in order to find said value. Considering that it should be a single value for one to eight agents, it makes sense to perform these tests with eight agents - which represent the highest load on AgentService. Considering the results obtained in Table 5.6, it was decided to start the tests at 10 messages per second. The ideal frequency should not only allow for the continuous use without errors but also not represent significant overhead in terms of processing time, i.e. five minutes of messages should be handled in not more than five minutes. The results of these tests are detailed in Table 5.7:

Table 5.7: AgentService Endurance Test

nAgents	nMessages	Messages/second	Errors	Time (minutes)	Effective Messages/second /agent	Effective messages/second
8	18000	10	0	29,38282557	10,21004595	81,68036759
8	21600	12	0	30,0314826	11,98742013	95,89936105
8	23400	13	0	29,98079165	13,00832892	104,0666314
8	25200	14	0	30,43242599	13,80106861	110,4085489
8	27000	15	ERROR	ERROR	ERROR	ERROR

Considering the obtained results, regarding the maximum frequency of messages to be used on the DM with any number of airplanes, it was concluded that 10 messages per second is a safe choice. The tests with 11 through 14 messages per second were also successful, however considering that the test with 15 messages per second was unsuccessful, the value 10 represents a

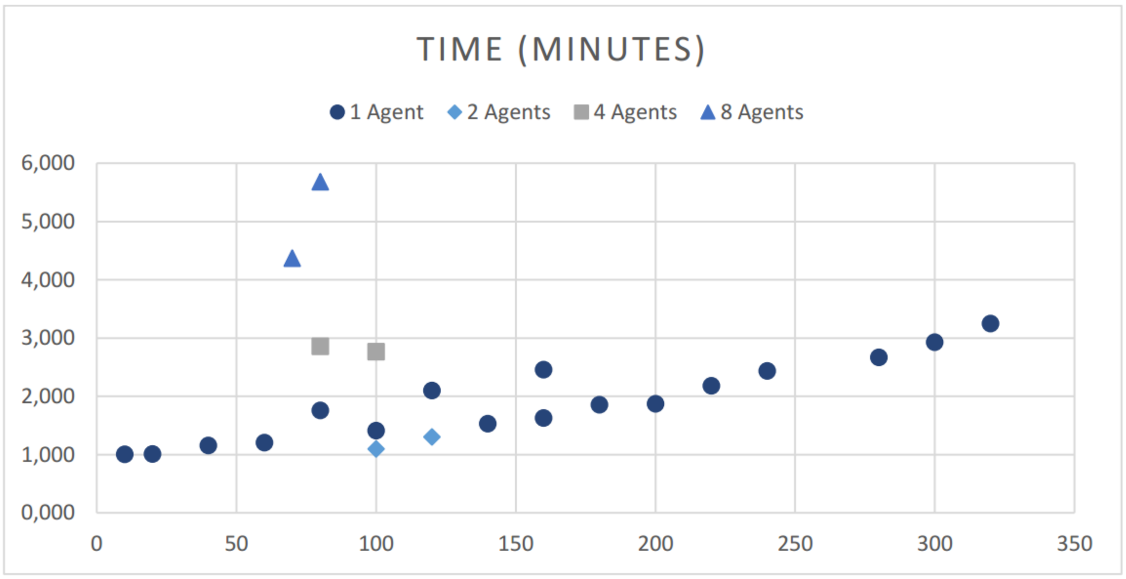


Figure 5.10: Time per Test

valid option with a reasonable margin of error. Besides, with 14 messages per second, the effective rate of messages was 13.801, which is slightly lower than 14, meaning that the processing overhead starts to become too high. As expected, the chosen value of 10 messages per second per agent, which translates to 80 global messages per second is close to the maximum message rate obtained in Table 5.4 for one agent, meaning that the number of agents doesn't have significant impact over AgentService's maximum capability - i.e. it is roughly the same to send 10 messages to one agent as it would be to send 1 message to ten agents. Unlike in the previous tables, the final entry was marked with *ERROR* after a single failed attempt - considering that this test was meant to find reliable message frequencies, additional test runs were not performed in order to achieve the rate of 15 messages/second/agent.

All the AgentService tests were performed on a machine with the following specifications:

- CPU: Intel Core i5 650 @ 3.20 GHz
- RAM: 8GB Single-Channel DDR3 @ 1333 MHz
- HDD: 7200RPM Seagate SATA
- OS: Windows 7 Professional 64-bit Service Pack 1

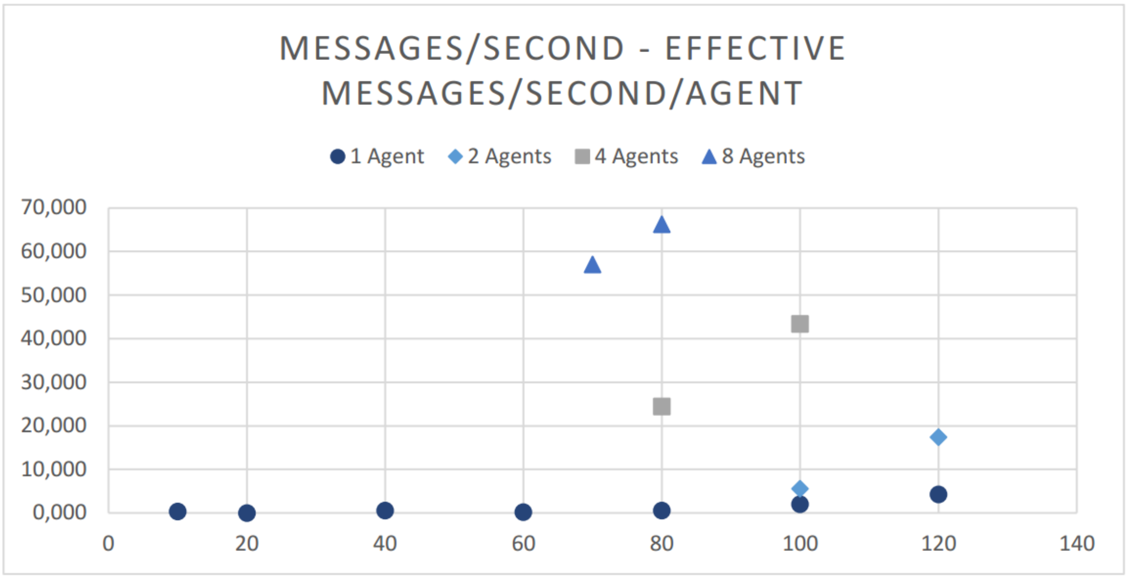


Figure 5.11: Difference Between Expected Message Rate and Effective Message Rate

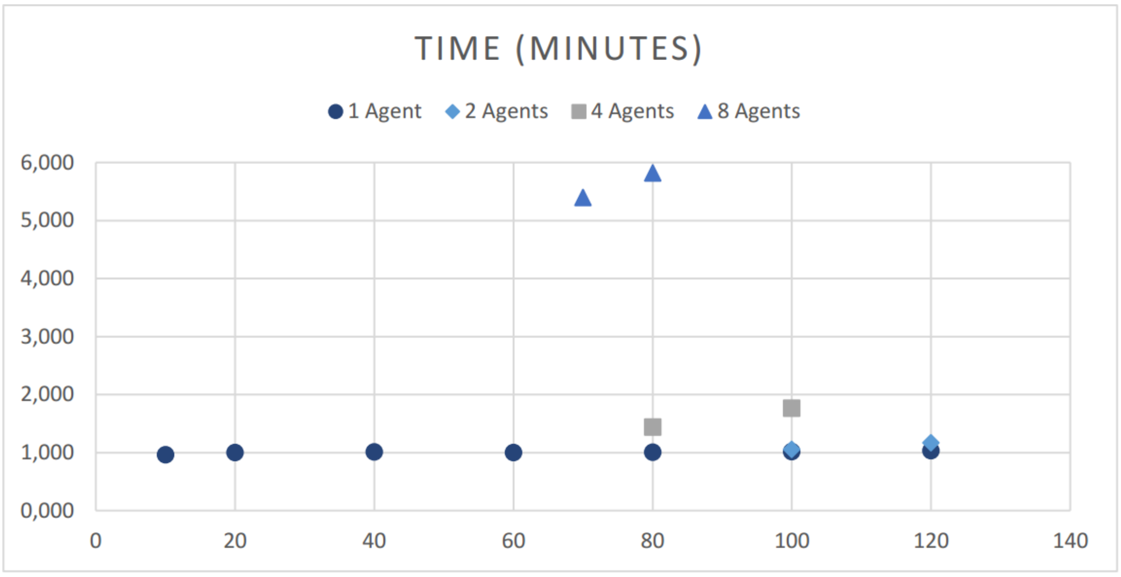


Figure 5.12: Time per Test

Chapter 6

Conclusions & Future Work

The chapter at hand includes a summary of the results and conclusions achieved throughout the development of this thesis, as well as the importance of the produced work, limitations of the DM and possibilities for future work.

In Main Goals & Research Questions, two research questions are stated, regarding the work which was planned ahead. At the present stage, they can be answered: **Is it possible to create an environmental disturbances manager, and effectively integrate it within the existing platform?** Yes. The Environmental Disturbances Manager was implemented and effectively integrated within The Platform, including the main features which were planned.

Is it feasible to introduce a generic interface for external disaster simulators into the system? Unfortunately, due to alterations in the Work Plan, this question can't be given an absolute answer since such interface was not developed (details in Final Work Overview). However, due to the modular structure and current feature set of the DM, it is considered most likely possible.

With the completion of the Disturbances Manager, The Platform became even more intricate. Future users will be able to take advantage of the DM in order to develop more realistic and complex vehicular missions, for example regarding the testing of search methods for the center of a forest fire - allowing for the acquisition of knowledge which could be directly applied in real-world firefighting. The possibilities for uses of The Platform were quite broad before the DM, which added an extra layer of potential to it.

It was found that FSX is indeed a very complete simulator, and serves as a solid basis for The Platform. Despite its age, it shows remarkable potential for additions, modifications and upgrades through its extensively-documented SDK. However complete, the documentation isn't the most user-friendly, which originated some difficulty throughout the development of the DM's features. The Platform was also rather interesting to work with; having multiple and diverse components, the understanding of its global set of functionalities took some time, but proved useful in the integration of the DM therein.

From the beginning to the end of the development of this thesis, many hardships arose and were overcome. The developed work is considered successful and proved to be rather challenging, yet rewarding.

6.1 Limitations

The limitation of the implemented Disturbances Manager which is considered most significant refers to AgentService and its modest capacity to handle large loads, in terms of the rate of messages. This reflects upon the Disturbances Manager as a limitation to potential future work - the implementation of different update frequencies of the vehicles' sensors, as described in AgentService Stress Test.

6.2 Future Work

Multiple extensions and improvements to the developed Disturbances Manager were considered throughout the development of the dissertation at hand:

- Implementation of a generic interface for external co-simulators, as was initially planned, and mentioned in Work Plan;
- Possibility for changing the specification of disturbances during simulation - besides the specification of the disturbances previous to their deployment, it would be a welcome addition to change the details of disturbances while they are being simulated. As an example, it could be useful to move an active disturbance to a different area of the map;
- More time and space evolution functions could be implemented, as to provide the means for simulating more diverse disturbances;
- The introduction of specific update rates for each sensor;
- More detailed in-game representation of disturbances. As described in SimConnect, the version of SimConnect does not allow for the programmatic deployment of Special Effects in the simulation. By migrating The Platform to Prepar3D, it would become possible to use this feature of the updated SimConnect, among others.

Bibliography

[IEE, 2010] (2010). IEEE standard for modeling and simulation (M&S) high level architecture (HLA)– framework and rules. *IEEE Std 1516-2010 (Revision of IEEE Std 1516-2000)*, pages 1–38.

[A. Finney, 2006] A. Finney, M. (2006). An overview of flammap fire modeling capabilities. In *Fuels Management-How to Measure Success: Conference Proceedings*.

[Brodie et al., 2007] Brodie, K., Fettes, D., Harte, B., and Schmid, R. (2007). Structural terms including fault rock terms. *Recommendations by the IUGS Subcommission*.

[Buscarino et al., 2015] Buscarino, A., Famoso, C., Fortuna, L., Frasca, M., and Xibilia, M. G. (2015). Complexity in forest fires: From simple experiments to nonlinear networked models. *Communications in Nonlinear Science and Numerical Simulation*, 22(1-3):660–675.

[Camara, 2013] Camara, Á. (2013). Controlo de tráfego aéreo usando o microsoft flight simulator x. Master's thesis, Department of Informatics Engineering, University of Coimbra, Coimbra, Portugal.

[Câmara et al., 2014] Câmara, Á., Silva, D. C., Abreu, P. H., and Oliveira, E. (2014). Comparing a centralized and decentralized multi-agent approaches to air traffic control. In *Proceedings of the 28th European Simulation and Modelling Conference (ESM'2014), October 22-24, 2014, Porto, Portugal*, pages 189–193.

[Chen et al., 2006] Chen, D., Turner, S. J., and Cai, W. (2006). A framework for robust hla-based distributed simulations. In *20th Workshop on Principles of Advanced and Distributed Simulation (PADS'06)*, pages 183–192.

[CRED, 2015] CRED (2015). The Human Cost of Natural Disasters 2015: Global Perspective. Technical report, Centre for Research on the Epidemology of Disasters (CRED).

[Dahmann et al., 1998] Dahmann, J. S., Kuhl, F., and Weatherly, R. (1998). Standards for Simulation: As Simple As Possible But Not Simpler The High Level Architecture For Simulation. *SIMULATION*, 71(6):378–387.

[Drabek, 1970] Drabek, T. E. (1970). Methodology of studying disasters. *American Behavioral Scientist*, 13(3):331–343.

[Filippi et al., 2011] Filippi, J.-B., Bosseur, F., Pialat, X., Santoni, P.-A., Strada, S., and Mari, C. (2011). Simulation of Coupled Fire/Atmosphere Interaction with the MesoNH-ForeFire Models. *Journal of Combustion*, 2011:1–13.

[Finney and Station-Ogden, 1998] Finney, M. and Station-Ogden, R. M. R. (1998). *FARSITE, Fire Area Simulator–model development and evaluation*. Number no. 4 in Research paper RMRS. U.S. Dept. of Agriculture, Forest Service, Rocky Mountain Research Station.

[Gewin, 2016] Gewin, V. (2016). Data sharing: An open mind on open data. *Nature*, 529(7584):117–119.

[Henri Balbi et al., 2009] Henri Balbi, J., Morandini, F., Silvani, X., Filippi, J. B., and Rinieri, F. (2009). A Physical Model for Wildland Fires. *Combustion and Flame*, 156(12):2217–2230.

[IEDC, 2016] IEDC (2016). Types of disasters. Last Accessed on 2016-11; Available online at http://restoreyoureconomy.org/disaster-overview/types-of-disasters/.

[Ivanilova, 1985] Ivanilova, T. (1985). Set probability identification in forest fire simulation. *Annual Review in Automatic Programming*, 12(Part 2):185 – 188. Systems Analysis and Simulation 1985.

[MAMAOT, 2015] MAMAOT (2015). Áreas ardidas anuais por tipo de ocupação do solo. Last Accessed on 2016-11; Available online at http://www.icnf.pt/portal/florestas/dfci/Resource/doc/estat/area-ardida-1996-a-2014.

[Martin, 2017] Martin, L. (2017). PREPAR3D SDK Overview. Last Accessed on 2017-05-20; Available online at http://www.prepar3d.com/SDKv4/sdk/sdk_overview.html.

[Microsoft, 2008a] Microsoft (2008a). Mission creation. Last Accessed on 2017-06; Available online at https://msdn.microsoft.com/en-us/library/cc526963.aspx.

[Microsoft, 2008b] Microsoft (2008b). Simconnect. Last Accessed on 2017-06; Available online at https://msdn.microsoft.com/en-us/library/cc526983.aspx.

[Microsoft, 2008c] Microsoft (2008c). Special effects tool. Last Accessed on 2017-02; Available online at https://msdn.microsoft.com/en-us/library/cc526969.aspx.

[NG, 2017] NG (2017). GIS (geographic information system) - National Geographic Society. Last Accessed on 2017-01; Available online at http://www.nationalgeographic.org/encyclopedia/geographic-information-system-gis/.

[NICIF and Lourenço, 1994] NICIF and Lourenço, L. (1994). Risco de incêndio florestal em Portugal Continental. *Colectâneas Cindínicas*, 1.

[NIDM, 2014] NIDM (2014). Types of environmental disasters. Last Accessed on 2016-11; Available online at http://nidm.gov.in/easindia2014/err/pdf/themes_issue/env/types.pdf.

[NOAA, 2017a] NOAA (2017a). Billion-dollar weather and climate disasters. Last Accessed on 2016-11; Available online at https://www.ncdc.noaa.gov/billions.

[NOAA, 2017b] NOAA (2017b). Billion-dollar weather and climate disasters: Table of events. Last Accessed on 2016-11; Available online at https://www.ncdc.noaa.gov/billions/events/US/1980-2017.

[Pereira and Rossetti, 2012] Pereira, J. L. F. and Rossetti, R. J. F. (2012). An integrated architecture for autonomous vehicles simulation. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, SAC '12. ACM.

[Quarantelli, 1992] Quarantelli, E. L. (1992). The environmental disasters of the future will be more and worse but the prospects is not hopeless. Technical report.

[Reid and Powers, 2000] Reid, M. R. and Powers, E. I. (2000). An Evaluation of the High Level Architecture (HLA) as a Framework for NASA Modeling and Simulation.

[Rodrigues et al., 2015] Rodrigues, C., Silva, D. C., Rossetti, R. J. F., and Oliveira, E. (2015). Distributed flight simulation environment using flight simulator x. In *Proceedings of the 10th Iberian Conference on Information Systems and Technologies, June 17-20 2015, Águeda, Portugal*, pages 1293–1297.

[Santos, 2010] Santos, A. (2010). Autonomous Intelligent Vehicle Adaptation and Performance Analysis in Flight Simulator X. Master's thesis, Faculty of Engineering, University of Porto, Porto, Portugal.

[Santos, 2012] Santos, W. (2012). 26 Weather APIs, 12 Support JSON | ProgrammableWeb. Last Accessed on 2017-02; Available online at https://www.programmableweb.com/news/26-weather-apis-12-support-json/2012/01/11.

[Silva, 2011] Silva, D. C. (2011). *Cooperative Multi-Robot Missions: Development of a Platform and a Specification Language*. PhD thesis, Faculty of Engineering, University of Porto, Porto, Portugal.

[Silva et al., 2017] Silva, D. C., Abreu, P. H., Reis, L. P., and Oliveira, E. (2017). Development of Flexible Languages for Scenario and Team Description in Multi-Robot Missions. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 31(1):69–86.

[Silva et al., 2016] Silva, D. C., Reis, P. H. A. L. P., and Oliveira, E. (2016). Development of a Flexible Language for Disturbance Description for Multi-Robot Missions. *Journal of Simulation*, 10(3):166–181.

[Silva, 2008] Silva, R. (2008). Communication Layer in an Aviation Simulator. Master's thesis, Faculty of Engineering, University of Porto, Porto, Portugal.

[Sousa, 2010] Sousa, P. D. (2010). Autonomous Air Traffic Control for Intelligent Vehicles using Microsoft Flight Simulator X. Master's thesis, Faculty of Engineering, University of Porto, Porto, Portugal.

[Sousa et al., 2010] Sousa, P. D., Silva, D. C., and Reis, L. P. (2010). Air Traffic Control with Microsoft Flight Simulator X. In Álvaro Rocha, Sexto, C. F., Reis, L. P., and Cota, M. P., editors, *Actas de la 5^a Conferencia Ibérica de Sistemas y Tecnologías de Información (CISTI 2010), June 16–19 2010, Santiago de Compostela, Spain*, pages 378–383. in Portuguese (original title: Controlo de Tráfego Aéreo com o Microsoft Flight Simulator X).

[Stevenson et al., 1975] Stevenson, A., Schermerhorn, D., and Miller, S. (1975). Simulation of southern california forest fires. *Symposium (International) on Combustion*, 15(1):147–155.

[USDA, 2015] USDA (2015). FARSITE. Last Accessed on 2017-03; Available online at https://www.firelab.org/project/farsite.

[USGS, 2016] USGS (2016). USGS: Volcano Hazards Program. Last Accessed on 2017-02; Available online at https://volcanoes.usgs.gov/vhp/hazards.html.

[Wells, 2008] Wells, G. (2008). The rothermel Fire-Spread Model: Still running Like a champ.

[Wong, 1998] Wong, V. (1998). Flight Simulation (Department of Computing | Imperial College London).

Appendix A

DDL XML File

A.1 Sample Disturbance Specification File

```xml
<?xml version="1.0" encoding="UTF-8"?>
<disturbances xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="dcs:disturbance disturbances_joao.xsd" xmlns="
    dcs:disturbance">
  <disturbance id="d0" disturbanceType="hurricane" origin="ManMade" seriousness="2"
      >
    <denomination>Fire Number One</denomination>
    <distLocation placing="random">
      <area id="l1">
        <denomination>Forest Area</denomination>
        <medium air="false" land="true" water="false" underwater="false" />
        <availability available="always" />
        <circle>
          <minAlt measured="agl" lengthUnit="Foot">0</minAlt>
          <maxAlt measured="agl" lengthUnit="Foot">1</maxAlt>
          <center>
      <latitude>48° 21’ 0.50’’ N</latitude>
      <longitude>122° 40’ 18.91’’ W</longitude>
          </center>
          <radius lengthUnit="Foot">1500</radius>
        </circle>
      </area>
    </distLocation>
    <distAvailability start="scheduled" end="unspecified">
      <starting>
        <timePoint countingFrom="simulationStart">
          <after>PT0H00M20S</after>
        </timePoint>
      </starting>
    </distAvailability>
    <distMobility mobilityType="Stationary" constrainedToArea="false">
    </distMobility>
    <component name="Fire0">
      <medium air="true" land="false" water="false" underwater="false" />
      <distSize growthType="fixed">
        <initialSize shape="paralelogram">
          <dimensions>
            <width lengthUnit="Foot">16000</width>
            <length lengthUnit="Foot">25000</length>
            <height lengthUnit="Foot">30</height>
          </dimensions>
        </initialSize>
      </distSize>
      <distDetection>
        <sense sensorType="CO2"/>
        <generatedReading type="Float" startValue="50" minLimitValue="0"
            maxLimitValue="100" growthFunction="linear"/>
        <coefficient>5</coefficient>
      </distDetection>
    </component>
  </disturbance>
  <disturbance id="d1" disturbanceType="forestFire" origin="ManMade" seriousness="2
      ">
    <denomination>Fire Number Two</denomination>
    <distLocation placing="random">
      <area id="l1">
        <denomination>Forest Area</denomination>
        <medium air="false" land="true" water="false" underwater="false" />
        <availability available="always" />
        <circle>
          <minAlt measured="agl" lengthUnit="Foot">0</minAlt>
          <maxAlt measured="agl" lengthUnit="Foot">1</maxAlt>
          <center>
      <latitude>48° 21’ 0.50’’ N</latitude>
      <longitude>122° 45’ 18.91’’ W</longitude>
          </center>
          <radius lengthUnit="Foot">1500</radius>
        </circle>
      </area>
    </distLocation>
    <distAvailability start="scheduled" end="unspecified">
      <starting>
        <timePoint countingFrom="simulationStart">
          <after>PT0H10M0S</after>
        </timePoint>
      </starting>
    </distAvailability>
    <distMobility mobilityType="Stationary" constrainedToArea="false">
    </distMobility>
    <component name="Fire1">
```

```xml
      <medium air="true" land="false" water="false" underwater="false" />
      <distSize growthType="grow">
        <initialSize shape="sphere">
          <dimensions>
            <width lengthUnit="Foot">8000</width>
            <length lengthUnit="Foot">8000</length>
            <height lengthUnit="Foot">15</height>
          </dimensions>
        </initialSize>
      </distSize>
      <distDetection>
        <sense sensorType="CO2"/>
        <generatedReading type="Float" startValue="10.0" minLimitValue="0.0"
            maxLimitValue="100.0" growthFunction="constant"/>
    <coefficient>0</coefficient>
      </distDetection>
    </component>
  </disturbance>
</disturbances>
```

DDL XML File

Appendix B

TDL XML File

B.1 Sample Section of a Team Specification File

```xml
<agent id="a1" agentTypeID="y1">
  <name>Lusitania</name>
  <description>Primeiro agente a voar</description>
  <initialLocation baseID="b1" locationID="p01">
    <heading headingType="True">180</heading>
  </initialLocation>
  <state>
    <fuel volumeUnit="Litre">2.5</fuel>
    <lights cabin="false" logo="false" wing="false" recognition="false" panel="
        false" strobe="false" taxi="false" landing="false" beacon="false" nav="
        false" />
    <doorsOpen>false</doorsOpen>
    <gearDown>true</gearDown>
    <flapsHandlePosition>3</flapsHandlePosition>
    <rudder>-0.01334</rudder>
    <aileron>-0.1213</aileron>
    <elevator>0.01223</elevator>
  </state>
  <simulatedAgent>
    <tailNumber>DCS01</tailNumber>
  </simulatedAgent>
  <realAgent>
    <communication>
      <controlFrequency frequencyUnit="MegaHertz">1245.75</controlFrequency>
      <activeFrequency frequencyUnit="MegaHertz">121.2</activeFrequency>
      <standByFrequency frequencyUnit="MegaHertz">124.5</standByFrequency>
    </communication>
    <ATC>
      <callSign>Air One</callSign>
      <tailNumber>NLS01</tailNumber>
      <name>Lusitania</name>
    </ATC>
  </realAgent>
  <payloads>
    <payload name="PayloadCenter">
      <sensors>
        <sensor sensorType="CO2" id="s1_a1">
          <dimensions>
            <width lengthUnit="Centimeter">20</width>
            <length lengthUnit="Centimeter">18</length>
            <height lengthUnit="Centimeter">14</height>
          </dimensions>
          <weight massUnit="Kilogram">4</weight>
          <operatingRequirements>
            <temperature>-40°C - 75°C</temperature>
            <humidity>0 - 99%</humidity>
            <voltage>3 - 5.5 vdc</voltage>
            <current>10 mA</current>
            <powerConsumption>&lt;100mW</powerConsumption>
          </operatingRequirements>
          <specifications>
            <specification name="Detects">CO2</specification>
            <specification name="Method">Infrared</specification>
            <specification name="Range">0-2000ppm</specification>
            <specification name="Accuracy">5%</specification>
            <specification name="Resolution">10ppm</specification>
            <specification name="Output Range">0-22 mA</specification>
            <specification name="Response Time">5 ms</specification>
          </specifications>
          <sensorReadingType type="numeric" unit="ppmv"/>
        </sensor>
        <sensor sensorType="IRCameraIntensity" id="s2_a1">
          <dimensions>
            <width lengthUnit="Centimeter">30</width>
            <length lengthUnit="Centimeter">35</length>
            <height lengthUnit="Centimeter">30</height>
          </dimensions>
          <weight massUnit="Kilogram">2</weight>
          <operatingRequirements>
            <temperature>-15°C - 90°C</temperature>
            <humidity>0 - 99%</humidity>
            <voltage>3 - 5.5 vdc</voltage>
            <current>25 mA</current>
            <powerConsumption>&lt;125mW</powerConsumption>
          </operatingRequirements>
            <specifications>
              <specification name="Detects">Generic</specification>
            </specifications>
            <sensorReadingType type="numeric" unit="X"/>
        </sensor>
      </sensors>
      <cargo type="water" quantity="180"/>
    </payload>
  </payloads>
</agent>
```

TDL XML File

Appendix C

XML Databases

C.1 Disturbances

```xml
<disturbances>
  <disturbanceCategory id="C01" name="Land">
    <disturbance id="L01" name="landslide" origin="Natural">
      <sense sensorType="CameraIntensity"/>
      <sense sensorType="Microphone"/>
      <seriousnessDegree applicable="1"/>
      <growthRate applicable="1"/>
    </disturbance>
    <disturbance id="L02" name="earthquake" origin="Natural">
      <sense sensorType="CameraIntensity"/>
      <seriousnessDegree applicable="1"/>
      <growthRate applicable="1"/>
    </disturbance>
    <disturbance id="L03" name="flood" origin="Natural">
      <sense sensorType="CameraIntensity"/>
      <seriousnessDegree applicable="1"/>
      <growthRate applicable="1"/>
    </disturbance>
    <disturbance id="L04" name="volcano" origin="Natural">
      <sense sensorType="CameraIntensity"/>
      <sense sensorType="IRCameraIntensity"/>
      <sense sensorType="Temperature">
        <emitsReading type="float" startValue="X" minValue="Y" maxValue="Z"/>
      </sense>
      <sense sensorType="Microphone"/>
      <seriousnessDegree applicable="1"/>
      <growthRate applicable="1"/>
    </disturbance>
    <disturbance id="L06" name="carAccident" origin="ManMade">
      <sense sensorType="CameraBinary"/>
      <seriousnessDegree applicable="1"/>
      <growthRate applicable="0"/>
    </disturbance>
    <disturbance id="L07" name="trainAccident" origin="ManMade">
      <sense sensorType="CameraBinary"/>
      <seriousnessDegree applicable="1"/>
      <growthRate applicable="0"/>
    </disturbance>
    <disturbance id="L08" name="fleeingPerson" origin="ManMade">
      <sense sensorType="CameraBinary"/>
      <sense sensorType="Microphone"/>
      <seriousnessDegree applicable="1"/>
      <growthRate applicable="0"/>
    </disturbance>
    <disturbance id="L09" name="lostAnimal" origin="ManMade">
      <sense sensorType="CameraBinary"/>
      <sense sensorType="Microphone"/>
      <seriousnessDegree applicable="1"/>
      <growthRate applicable="0"/>
    </disturbance>
  </disturbanceCategory>
  <disturbanceCategory id="C02" name="Sea">
    <disturbance id="S01" name="strongWaves" origin="Natural">
      <sense sensorType="CameraIntensity"/>
      <sense sensorType="Microphone"/>
      <seriousnessDegree applicable="1"/>
      <growthRate applicable="1"/>
    </disturbance>
    <disturbance id="S02" name="tsunami" origin="Natural">
      <sense sensorType="CameraIntensity"/>
      <sense sensorType="Microphone"/>
      <seriousnessDegree applicable="1"/>
      <growthRate applicable="1"/>
    </disturbance>
    <disturbance id="S03" name="oilSpill" origin="ManMade">
      <sense sensorType="CameraIntensity"/>
      <seriousnessDegree applicable="1"/>
      <growthRate applicable="0"/>
    </disturbance>
    <disturbance id="S04" name="chemicalSpill" origin="ManMade">
      <sense sensorType="CameraIntensity"/>
      <sense sensorType="Chemical"/>
      <seriousnessDegree applicable="1"/>
      <growthRate applicable="0"/>
    </disturbance>
    <disturbance id="S05" name="boatAccident" origin="ManMade">
      <sense sensorType="CameraBinary"/>
      <seriousnessDegree applicable="1"/>
      <growthRate applicable="0"/>
    </disturbance>
    <disturbance id="S06" name="lostPerson" origin="ManMade">
      <sense sensorType="CameraBinary"/>
      <seriousnessDegree applicable="1"/>
      <growthRate applicable="0"/>
    </disturbance>
    <disturbance id="S07" name="hydrothermalVent" origin="Natural">
      <sense sensorType="CameraIntensity"/>
      <sense sensorType="Temperature"/>
      <sense sensorType="Microphone"/>
      <seriousnessDegree applicable="1"/>
      <growthRate applicable="0"/>
    </disturbance>
    <disturbance id="S08" name="schoolingFish" origin="Natural">
      <sense sensorType="CameraIntensity"/>
      <sense sensorType="Chemical"/>
      <seriousnessDegree applicable="1"/>
      <growthRate applicable="0"/>
    </disturbance>
  </disturbanceCategory>
  <disturbanceCategory id="C03" name="Fire">
    <disturbance id="F01" name="forestFire" origin="Any">
      <sense sensorType="CameraIntensity"/>
      <sense sensorType="IRCameraIntensity"/>
      <sense sensorType="Temperature">
        <emitsReading type="float" startValue="X" minValue="Y" maxValue="Z"/>
      </sense>
      <sense sensorType="CO2"/>
      <seriousnessDegree applicable="1"/>
      <growthRate applicable="1"/>
    </disturbance>
    <disturbance id="F02" name="buildingsFire" origin="ManMade">
      <sense sensorType="CameraIntensity"/>
      <sense sensorType="IRCameraIntensity"/>
      <sense sensorType="Temperature">
        <emitsReading type="float" startValue="X" minValue="Y" maxValue="Z"/>
      </sense>
      <sense sensorType="CO2"/>
      <seriousnessDegree applicable="1"/>
      <growthRate applicable="1"/>
    </disturbance>
  </disturbanceCategory>
  <disturbanceCategory id="C04" name="Storm">
    <disturbance id="S01" name="heavyRain" origin="Natural">
      <sense sensorType="CameraIntensity"/>
      <sense sensorType="HumiditySensor"/>
      <sense sensorType="AirPressure"/>
      <sense sensorType="WindSpeed"/>
      <sense sensorType="Microphone"/>
      <seriousnessDegree applicable="1"/>
      <growthRate applicable="1"/>
    </disturbance>
    <disturbance id="S02" name="thunderstorm" origin="Natural">
      <sense sensorType="CameraIntensity"/>
      <sense sensorType="HumiditySensor"/>
      <sense sensorType="AirPressure"/>
      <sense sensorType="WindSpeed"/>
      <sense sensorType="Microphone"/>
      <sense sensorType="ElectricField"/>
      <sense sensorType="MagneticField"/>
      <seriousnessDegree applicable="1"/>
      <growthRate applicable="1"/>
    </disturbance>
    <disturbance id="S03" name="hurricane" origin="Natural">
      <sense sensorType="CameraIntensity"/>
      <sense sensorType="AirPressure"/>
      <sense sensorType="WindSpeed"/>
      <sense sensorType="Microphone"/>
      <seriousnessDegree applicable="1"/>
      <growthRate applicable="1"/>
    </disturbance>
  </disturbanceCategory>
  <disturbanceCategory id="C05" name="General">
    <disturbance id="G01" name="bugInfestation" origin="Natural">
      <sense sensorType="CameraIntensity"/>
      <seriousnessDegree applicable="1"/>
      <growthRate applicable="1"/>
    </disturbance>
    <disturbance id="G02" name="extremeTemperature" origin="Natural">
      <sense sensorType="Temperature">
        <emitsReading type="float" startValue="X" minValue="Y" maxValue="Z"/>
      </sense>
      <seriousnessDegree applicable="1"/>
      <growthRate applicable="1"/>
    </disturbance>
    <disturbance id="G03" name="coldWave" origin="Natural">
      <sense sensorType="Temperature">
        <emitsReading type="float" startValue="X" minValue="Y" maxValue="Z"/>
      </sense>
      <seriousnessDegree applicable="1"/>
      <growthRate applicable="1"/>
    </disturbance>
    <disturbance id="G04" name="heatWave" origin="Natural">
      <sense sensorType="Temperature"/>
        <emitsReading type="float" startValue="X" minValue="Y" maxValue="Z"/>
      <seriousnessDegree applicable="1"/>
      <growthRate applicable="1"/>
    </disturbance>
    <disturbance id="G05" name="chemicalContamination" origin="ManMade">
```

```xml
      <sense sensorType="ChemicalSensor"/>
      <seriousnessDegree applicable="1"/>
      <growthRate applicable="1"/>
    </disturbance>
    <disturbance id="G06" name="chemicalCloud" origin="ManMade">
      <sense sensorType="ChemicalSensor"/>
      <seriousnessDegree applicable="1"/>
      <growthRate applicable="1"/>
    </disturbance>
    <disturbance id="G07" name="underFire" origin="ManMade">
      <sense sensorType="CameraIntensity"/>
      <seriousnessDegree applicable="1"/>
      <growthRate applicable="1"/>
    </disturbance>
    <disturbance id="G08" name="buildingsInRuins" origin="ManMade">
      <sense sensorType="CameraBinary"/>
      <seriousnessDegree applicable="1"/>
      <growthRate applicable="0"/>
    </disturbance>
    <disturbance id="G09" name="radiation" origin="ManMade">
      <sense sensorType="GeigerCounter"/>
      <seriousnessDegree applicable="1"/>
      <growthRate applicable="0"/>
    </disturbance>
    <disturbance id="G10" name="underFire" origin="ManMade">
      <sense sensorType="CameraIntensity"/>
      <sense sensorType="Microphone"/>
      <seriousnessDegree applicable="1"/>
      <growthRate applicable="0"/>
    </disturbance>
  </disturbanceCategory>
</disturbances>
```

C.2 Sensors

```xml
<sensors>
  <sensor name="Temperature" readingType="numeric" unit="degreeCelcius"/>
  <sensor name="CO2" readingType="numeric" unit="ppmv"/>
  <sensor name="CO" readingType="numeric" unit="ppmv"/>
  <sensor name="CameraBinary" readingType="boolean"/>
  <sensor name="CameraIntensity" readingType="seriousness"/>
  <sensor name="IRCameraIntensity" readingType="seriousness"/>
  <sensor name="ChemicalSensor" readingType="numeric" unit="genericFloat"/>
  <sensor name="Microphone" readingType="numeric" unit="genericFloat"/>
  <sensor name="Chemical" readingType="numeric" unit="genericFloat"/>
  <sensor name="AirPressure" readingType="numeric" unit="genericFloat"/>
  <sensor name="WindSpeed" readingType="numeric" unit="genericFloat"/>
  <sensor name="GeigerCounter" readingType="numeric" unit="genericFloat"/>
  <sensor name="ElectricField" readingType="numeric" unit="genericFloat"/>
  <sensor name="MagneticField" readingType="numeric" unit="genericFloat"/>
</sensors>
```

Appendix D

XSD Additions

D.1 DDL

```
<xs:attributeGroup name="distOriginGroup">
  <xs:attribute name="seriousness">
    <xs:simpleType>
    <xs:restriction base="xs:integer">
      <xs:minInclusive value="0"/>
      <xs:maxInclusive value="5"/>
    </xs:restriction>
  </xs:simpleType>
</xs:attribute>

<xs:attribute name="origin" use="required">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="Natural"/>
      <xs:enumeration value="ManMade"/>
      <xs:enumeration value="Any"/>
    </xs:restriction>
  </xs:simpleType>
</xs:attribute>

<xs:attribute name="growthFunction" use="optional">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="linear"/>
      <xs:enumeration value="polynomial"/>
      <xs:enumeration value="exponential"/>
      <xs:enumeration value="logarithmic"/>
      <xs:enumeration value="root"/>
      <xs:enumeration value="sinusoid"/>
      <xs:enumeration value="custom"/>
    </xs:restriction>
  </xs:simpleType>
</xs:attribute>

<xs:element name="distDetection">
<xs:complexType>
  <xs:sequence>
  <xs:element ref="sense" maxOccurs="unbounded"/>
  <xs:element ref="generatedReading" minOccurs="1" maxOccurs="1"/>
  <xs:sequence>
    <xs:element ref="coefficient" maxOccurs="unbounded"/>
  </xs:sequence>
  </xs:sequence>
</xs:complexType>
</xs:element>

<xs:element name="generatedReading">
<xs:complexType>
  <xs:attributeGroup ref="generatedReadingAttributes"/>
  <xs:attributeGroup ref="growthFunctionGroup_optional"/>
  <xs:attributeGroup ref="functionVariableGroup"/>
</xs:complexType>
</xs:element>

<xs:attributeGroup name="generatedReadingAttributes">
<xs:attribute name="type" use="required">
  <xs:simpleType>
  <xs:restriction base="xs:string">
    <xs:enumeration value="Binary"/>
    <xs:enumeration value="Float"/>
  </xs:restriction>
  </xs:simpleType>
</xs:attribute>
<xs:attribute name="startValue" type="xs:string" use="optional"/>
<xs:attribute name="minLimitValue" type="xs:string" use="optional"/>
<xs:attribute name="maxLimitValue" type="xs:string" use="optional"/>
</xs:attributeGroup>
```

D.2 TDL

```
<xs:element name="sensorReadingType">
<xs:complexType>
  <xs:simpleContent>
  <xs:extension base="xs:string">
    <xs:attribute name="type" type="xs:string" use="required"/>
    <xs:attribute name="unit" type="xs:string" use="required"/>
  </xs:extension>
  </xs:simpleContent>
</xs:complexType>
</xs:element>
```