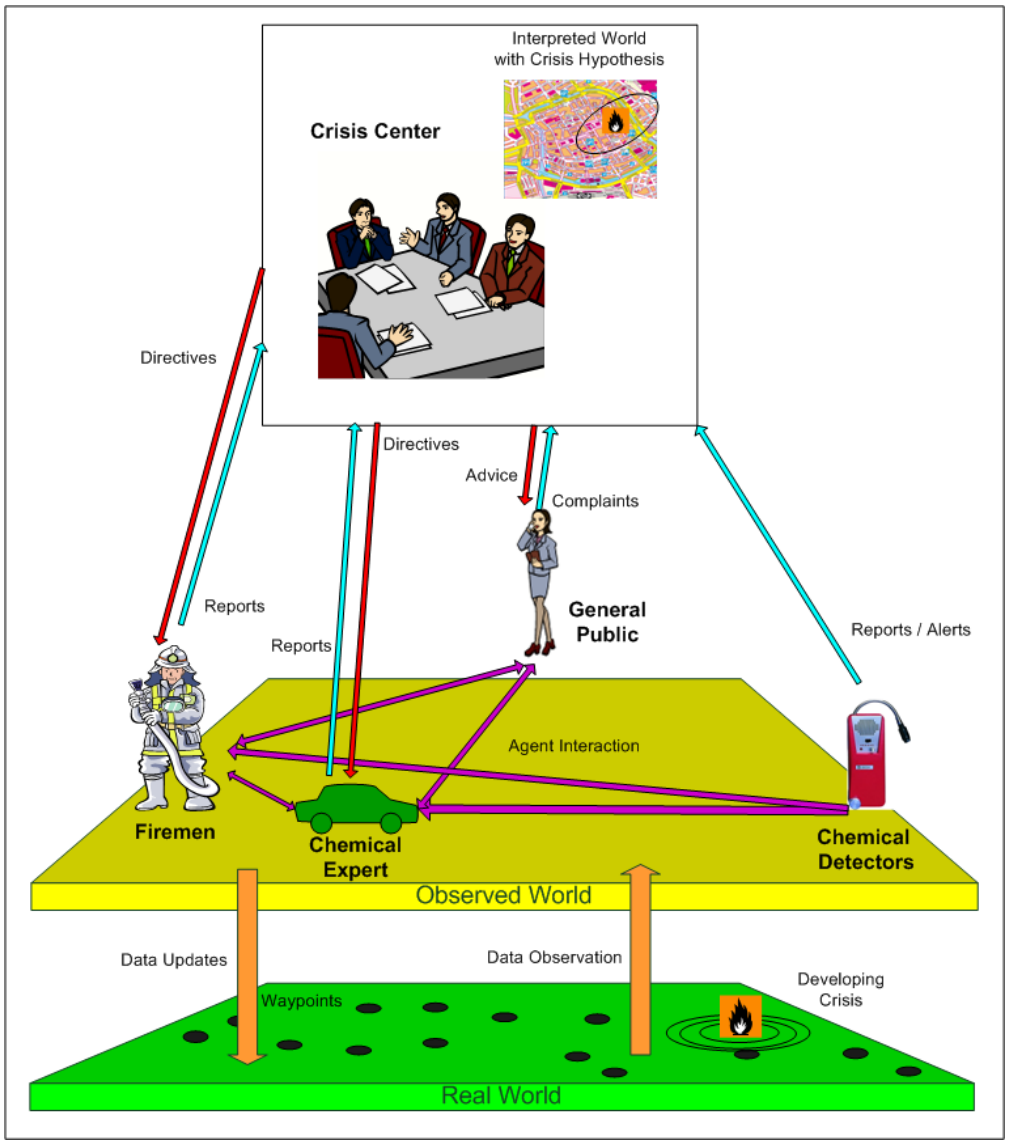
Thesis Report

MACSIM:

Multi-Agent Crisis Simulator, Interpreter and Monitor



Tom Benjamins, 1047191

November 5, 2006

Man-Machine Interaction Group
Faculty of Electrical Engineering,
Mathematics and Computer Science
Delft University of Technology,
The Netherlands
http://mmi.tudelft.nl

Delft University of Technology

Tom Benjamins

MACSIM:

Multi-**A**gent **C**risis **S**imulator, **I**nterpreter and **M**onitor

Thesis Report

November 5, 2006

Man-Machine Interaction Group
Faculty of Electrical Engineering,
Mathematics and Computer Science

Delft University of Technology,
The Netherlands

http://mmi.tudelft.nl

Abstract

MACSIM: Multi-Agent Crisis Simulator, Interpreter and Monitor
Tom Benjamins (1047191)
Man-Machine Interaction Group
Faculty of EEMCS
Delft University of Technology

Graduation Committee:
Dr. Drs. L.J.M. Rothkrantz, Dr. Ir. C.A.P.G. van der Mast, Dr. K. van der Meer

Major crisis incidents in the last couple of years show that there exist huge problems in the current practice of crisis management. It is a combination of failure in communication, failure in technology, failure in methodology, failure of management and finally failure of observation. It is therefore important that solutions are provided to prevent these kinds of incidents in the future. When a large incident should strike, the authorities are very interested in a system that can give possible hypotheses in an early stage of a crisis. Also there is a lot of room for improvement in the training of skills required for crisis management such as teamwork, coping with stress, filtering incoming messages and making the right command decisions based on the available information.

In search for a solution we tried to give an answer to the question to which extent it is possible to develop a Multi-Agent System and crisis environment supported by AI-reasoning to create a realistic simulation of a crisis and the corresponding crisis response by the authorities. By doing interviews with domain experts we came across the problems occurring in crisis management and requirements for a tool that crisis responders are waiting for.

The proposed solution is called MACSIM, which stands for Multi-Agent Crisis Simulator, Interpreter and Monitor. It is a multi-component disaster simulation and detection system that, if fully developed, is a tool that can be used for realistic simulation of crisis situations. It is used for training purposes to get a quick assessment in the first few minutes after the unfolding of an event. The input of the system is based on two data sources, detectors and reports from people inside the concerned area. The system consists of a simulation component, a multi-agent communication layer and a set of agents acting inside a virtual world.

The system is using JADE as a multi-agent platform for the agents to communicate. The Agents in MACSIM are able to reason about the phenomena they observe in the environment. They use a rule-based reasoning engine using Jess. The agents in the field observe the values; they are stored inside the reasoning engine and based on the a-priori knowledge represented in Jess rules in the reasoning engine, the agents make decisions. Either they make a decision on what action to perform in the world (Agents in the field) or they decide on a crisis hypothesis and appropriate response to the agents in the field (Crisis Center).

We tested the system with two example scenarios that implement everything that is currently possible with the system as far as showing simulation scenarios is concerned. Despite the fact that the currently implemented prototype is a proof of concept, the results of the two experiments are very promising because the system showed the expected behavior during the tests.

Keywords: Serious gaming, Artificial Intelligence, Crisis Simulation, JADE, Intelligent Agents, Jess

Acknowledgements

Scientific work is never done alone. The work for this thesis started in July 2005 and ended mid-August 2006, followed by the writing of this thesis in the following months. During this period of time I learnt a lot about myself and a lot from other people. In these few words I wish to thank everybody that helped me in completing this MSc project sucessfully.

First I'd like to thank the people that were directly involved in the MACSIM project at the MMI Lab of TU Delft. Leon Rothkrantz was my thesis supervisor and the one who conceived the first basic sketches of the MACSIM platform. Discussions with him were long, always fascinating and sometimes the discussion topic was even the actual project. Siska Fitrianie helped me during the final stages of the project with the inclusion of her message ontology and outlining the structure of this thesis document. Her patience and enthusiasm were a treat. And of course I want to thank Iulia Tatomir, whose components of 3MNews were used for visualisation. During the months we worked together I was given the opportunity to appreciate her as an inspiring colleague and, more importantly, as a remarkable human being. Thanks for that. Of the other people at MMI on the 12th floor I especially wish to thank Jan Misker for helping me with my first humble steps in JADE programming, and Zhenke Yang for providing information about the Jess environment.

Furthermore thanks to the domain experts that were interviewed for their time and patience. These are René Kuijper, head of the control room at DCMR, Richard van Haagen, chemical expert at DCMR, Leo Kooijman, project coordinator of Multi-Team and former head of crisis management at the Rotterdam fire department and finally Sjaak Seen, deputy chief of the Rotterdam fire department, southern district. Their first-hand knowledge was invaluable for the development of MACSIM and getting a basic idea of what kind of problems people working in crisis management come across. Additional useful research was provided by Jeroen van den Hoven, professor in Ethics and ICT at TU Delft and Freddie Kootstra, environmental researcher at TNO.

Finally I wish to thank my parents, my brother Steven, my cousin Ingrid, and my friends Vincent Sterk, Paul Klapwijk and Vincent de Lange for supporting me and now and then pointing out to me there was light at the end of the tunnel, when I needed it most. And last but not least, I wish to express my gratitude to Rick van 't Hoff and Dagmar Stadler, who during the last year gave me invaluable advice and support that helped me while doing this project. Without their support, life would have been much more difficult.

November 5, 2006,

Tom Benjamins

Contents

Chapter 1

Introduction

1.1 Motivation

This thesis is about Crisis Management. It is also about Crisis Simulation. Furthermore it's about Multi-Agent Systems and Using AI for crisis assessment. This list of terms indicates that the scope of the research being conducted in the period of June 2005 until august 2006 has been very wide. In this section we try to give a rationale and the outlines and problems that were the inspiration for our research efforts presented in our thesis work.

In the last couple of years, major incidents have occurred that shocked the world. The attacks on The World Trade Center in New York and the Pentagon in Washington, DC on September 11th 2001 are still fresh in the collective memory. When the first World Trade Center building collapsed on 9/11, on which the antenna for mobile communication was placed. With that antenna gone, there was no mobile telephone traffic possible, and no messages could be sent to the people in the second tower. If the information that was available on ground zero could have been sent to the policemen and firemen that were still inside the building, they could have decided to evacuate in time and they would still have been alive [KHBV$^+$04].

Closer to home, in Enschede, a fireworks disaster took place on June 13th 2000, in which an explosion of enormous amounts of heavy fireworks occurred as the result of a seemly innocent fire which ultimately eradicated an entire residential area. In the case of the Enschede explosion, lack of the local authorities to check permits and to check whether there were not too many pieces of heavy fireworks in the middle of a residential area in the first place [OdBE$^+$01].

Not too much later, on January 1st 2001, a fire in a small café in Volendam caused many deaths because of the absence of decent information about flight routes and communicational chaos during the response effort itself [Ald02].

During the fatal Space Shuttle Columbia flight on February 1st 2003 it was noticed that on reentering the earth orbit one of the wings sustained an unusual amount of heat because while leaving the orbit some of the protected tiles for heat cover were damaged. Enormous amounts of frictional heat were allowed to enter the structures of one of the wings, which eventually caused the entire structure of the spaceship to collapse, and resulting in the death of all 7 crewmembers of mission STS-107. Later on it was concluded that if that information had been distributed to mission control in time, a rescue effort had been still one of the possibilities and the crew could have been saved [Geh03]. This showed that some data that at first sight might be unimportant but can in hindsight contain the most essential information that could have prevented a lot of casualties. Another problem seems to be the difference of interpretation by different people. Sometimes the preliminary unfolding of a crisis event can be noticed by someone with an inexperienced eye (for instance a member of the general public), which leads to a report that contains essential information in hindsight, but is being put aside as useless because of the

lack of essential details that a trained expert-eye would have given, had he seen the same thing that the member of the general public saw [KBR05b], [vHBR06].

These example incidents show that there exist huge problems in the current practice of crisis management. It is a combination of failure in communication (people that should be communicating with each other but for some reason don't), failure in technology (communicational structures break down), failure in methodology (crisis plans that are not functioning properly), failure of management (management that refuses to acknowledge warnings) and finally failure of observation (information that was available was never noticed by the crisis response team).

It is therefore important that solutions are provided to prevent these kinds of incidents in the future. Naturally, scientific research in the field of Crisis Management and Response is "hot". Because of these huge crisis events the demand for tools that facilitate information sharing and more efficient communication during a crisis event increased significantly. Especially in the area of intelligent sharing of information there is a lot of research going on.

So far information technology has already been able to help the crisis response community. For instance, C2000, the communication system for all national crisis organizations, is in use for all emergency situations. Also the Fire department works with a knowledge-sharing tool called Multi-Team that enables the users to share all different types of knowledge efficiently during a crisis event. The United States Government Environmental Protection Agency (EPA) has developed a tool (ALOHA) which enables American government agencies to predict the effects of a certain gas dispersion on a certain American town or county (based on national Census data). These tools and efforts will be discussed in more detail in the Theory Chapter.

In the Man Machine Interaction department at TU Delft much research is being conducted in the field of crisis management and crisis response. Much of this research is performed within the DECIS (Delft Cooperation on Intelligent Systems) lab setting by Dr. Rothkrantz and his staff. DECIS is a project group of which the MMI department of TU Delft along with THALES, the University of Amsterdam and TNO are participants.

Two prominent projects within the DECIS setting are COMBINED and ICIS. The COMBINED project aims at creating Decision Support systems in a crisis situation. This is based on software- based intelligent support on a high level (Threat assessment and decision making) and on a low level (situation awareness and control). ICIS stands for Intelligent Collaborative Information Systems. The aim of ICIS is to develop better techniques for making complex information systems more intelligent and supportive in decision-making situations. Contemporary human decision-making (whether individual or organizational) has its limitations. The goal of ICIS is to develop new concepts and new techniques for intelligent support systems. It is believed that with those new concepts and techniques it is possible to dramatically improve performance. It is important to assure that the chaos in the environment does not lead to chaos in the decision making process. The ICIS project combines IT research and development with research in the field of human sciences. The human science component is required in order to better understand limitations to situation awareness and decision-making. This knowledge is then used to further enhance the proposed technological concepts and solutions (see http://www.icis.decis.nl).

Sometimes even information that at first sight seems trivial can contain the most vital tips when seen by the right person. The only problem is that in a lot of cases it's impossible to let every single bit of incoming data be observed by humans who know every thing about it. This is exactly why the use of AI-based expert systems has come up and expert systems are programmed to monitor data and perform intelligent data-mining in such a way that the users of a monitoring system will be informed whenever a potential anomaly might be at hand.

A lot of factories already have intelligent expert systems inside their facilities to quickly diagnose possible incidents based on sensory input that has been led through an expert system. The expert system

then gives early warnings in case of a problem and also gives suggestions for possible causes for the problem. This gives interesting possibilities for testing this principle on a larger scale, for instance in public space. It could be interesting for the fire service to have such a system at their disposal during fire incidents to give a quick assessment of a developing crisis situation.

In the case of a large-scale incident, such as a terrorist attack or a chemical release the authorities could also be very interested in a system that can give possible incident hypotheses in a very early stage of a crisis. Because the mayor, the crisis response staff and other officials want to make as well founded decisions as possible, it could be a great improvement for them if they could get possible scenario's based on real data to facilitate decision making. Sometimes humans with the required crisis management experience are not always available at the right time or location. Because it is also expected that decision-making situations will become more and more complex, there will be a growing need for intelligent systems, in particular decision support systems. Decision makers are confronted with environments that constantly change. Nevertheless it is important to stress here that such a system should be considered primarily as an advisory system, and that it should not be used as the only guideline for the decision process. The availability of real life experts will always be most important [KBR05b].

Not only the use of improved observational and information distribution techniques during the unfolding of a crisis event need to be considered, also the training of skills required for crisis management such as teamwork, coping with stress, filtering incoming information to get the right information out and making the right command decisions based on the available information. A significant increase of efficiency can be achieved when basic crisis response efforts are being trained at multiple levels regularly [KBR05a].

On the other hand, some disasters are very difficult to train in real life. Some disasters have never happened in real life and the effects are therefore very difficult to model in real life simulation. In these cases simulation in real life becomes unpractical or even impossible. Also Rothkrantz suggests in his paper about serious gaming that training is essential, but there are still a lot of problems with the current simulation efforts. The most prominent problems are [Rot06]:

- Not enough sense of reality with the contributors
- Scenarios are too rigid and cannot adapt to unexpected responses from contributors
- Simulating large real-life crises is expensive
- Simulation of large effects of a certain event, like the collapsing of a building are difficult to model realistically

Also Rothkrantz states that interactive simulation is a required tool for more effective crisis management. With the help of IT-tools it should become possible to create flexible simulation solutions, which can be easily adapted to an existing environment. Games provide tools for interactive simulation. The games currently used are performance-wise mature enough and are capable of providing a high level of presence, realism and validity to create a vast potential of simulation research efforts. Because of their maturity games prove that it is possible to model crisis situations in a realistic way. The only problem with games is that not for every crisis situation an off-the shelf simulation game is available. More flexibility is needed that can be offered by the simple use of simulation games that already exist.

During the development of crisis simulation software and intelligent components nevertheless some problems arise. Knowledge to model the properties and behavior of crisis responders is not explicit. Therefore research must be carried out to determine what kind of knowledge crisis response people use during the unfolding of a crisis. Also the distribution of simulation and communication data is difficult to model. During an event that happens those evens must be represented in such a way that it is not too difficult to model, but the representation should still contain enough information to make

it an accurate simulation. Finally the software that is needed to achieve simulations is in some cases already partly available, but the problem with this software is that it's either too expensive or closed source software. Therefore in this situation it is difficult to build new implementation ideas on top of existing software. This makes the option of developing our own software more attractive. One has the freedom of designing the software in such a way that above mentioned flexibility is guaranteed.

1.2 Problem Definition

In this section the main problems that are addressed in this thesis are outlined. The problem definition will be presented as a build-up consisting of four sub problems, converging into one summarizing problem definition that will be the focus point of this thesis.

1. ***For training, testing and research simulation is necessary. What kind of crsis situations and effects should be simulated?*** – If we have to simulate a crisis, what are the things that we want the participants of the simulation to see, what are the effects that should be generated inside data structures?

2. ***Can a system consisting of components (simulation, reasoning engines, agent actions) be designed as a first prototype in a proof of concept that will be the foundation for further iterative improvement?*** – Because a system that we have in mind is a very complex structured piece of software, it is absolutely vital that we are able to update simplified pieces of algorithm or components easily, without the need for completely redesigning the system.

3. ***How can we design a crisis simulator with a high sense of realism?*** – If we design a crisis simulator, the data used should not be nonsense. We have to make sure that the simluation presented in a program to be developed is not just some sort of fantasy, but is actually believable for the participants.

4. ***How to apply a Multi-Agent System for crisis management simulation?*** – Are the properties of a Multi-Agent System suited for the communicational structure required for crisis management? Is it possible to model emergent behavior into the agents' way of acting? This mens that agents in the field autonomously respond according to observations and higher levels of command must respond to this sometimes unexpected behaviour.

5. ***Can AI-reasoning help in simulating crisis management-related decision-making?*** – Which AI methods are applicable to model the decision making process in a crisis management and response environment?

To summarize these questions we can come to the following main question, which will be the formal problem definition of this thesis:

To which extent is it possible to develop a Multi-Agent System
and crisis environment supported by AI-reasoning
to create a realistic simulation of a crisis
and the corresponding crisis response by the authorities?

1.3 Solution

The proposed solution is called MACSIM, which stands for Multi-Agent Crisis Simulator, Interpreter and Monitor. It is based on a multi-component disaster simulation and detection system that, if fully developed, can be a starting point for development as a tool that is to be used for realistic simulation of real situations. Ideas for the exact implementation are retrieved from the scenario that DECIS made to describe the first 3 hours after a collision of two boats in the Rotterdam harbor [vdV04]. In this scenario a lot of hi-tech appliances that are currently available are integrated in an "every day reality of the future." Based on this scenario a description of a solution was made that can be the base for an intelligent crisis response system that can give advice on the nature of an incident and be a simulator for a crisis event.

The first step would be a system that can be used for training purposes to get a quick assessment in the first few minutes after the unfolding of an event. The input of the system should be based on two data sources:

Detectors - In an area, the air quality and other physical features like certain gas concentrations are constantly being monitored by the environmental agency.

Reports from people inside the area - People in the Netherlands can call the environmental agency to issue complaints about things they smell, hear or see in the surrounding area. Also experts that are at a crisis situation are capable of sending reports. All reports that are being sent are based on the observations, but also on the knowledge that those people have in advance.

These types of data come into the system that, based on a reasoning engine, indicates which hypothetical crisis situation might be at hand. Based on a graphical representation of the area (which is also available in a real crisis center) the incoming data can be represented and updated should new reports come in. The most probable crisis situation can be shown in a separate part of the screen and can change through the course of the crisis event, as new information comes available.

The proposed solution should be found in an open platform (preferably programmed in Java to facilitate modeling and rapid prototype development) that is designed in such a way that replaceable parts are being created. This because of the fact that this system will be far from complete during the first design-implementation iteration and when new versions of the system should be planned, components from the first version should be replaced and modified very easy without to many changes in other pieces of the code. In this way, new concepts and ideas that are not completely developed in the first prototype of this system can, as new thesis projects, be integrated into new versions of the system and replace and / or enhance the current implemented functionality and possibilities.

For now the system consists of the following components:

Simulation Component

In this component a simulation of the concerned area can be given. The physical properties on a location (x, y, z) on time t can be read, and in our first prototype this means for instance fire, gas dispersion and explosions. The gas dispersion formulae are also used inside commercial available software. The Gaussian Plume Model, to be exact. The simulation will be simplified to reduce computing complexity and design issues, because creating a scientifically valid simulation could be a full master project on its own, and as the purpose of this system was to create a design proof of concept, a slimmed-down version of the simulation will be available, to prevent the development from this component to overshadow the development of the reasoning engine.

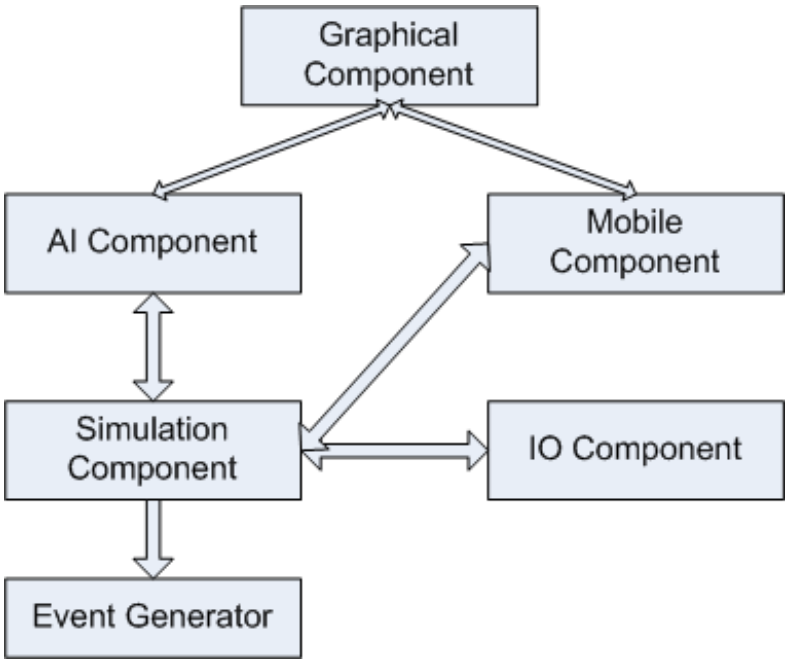


Figure 1.1: Global Overview of MACSIM components

Event Generator

This event generator should be able to generate crisis events, based on a dynamicscript. When the script unfolds, events are being launched and those events have their effect on the simulation environment. This event generator in the future should work in two ways: first it generates events from a predefined script, secondly it should be able to generate new events as the participants in the simulator respond to the system. Based on the time schedule it will be decided which ones of these two forms of scripting will be available in the first version of MACSIM.

Graphical Component

The graphical component of this program will consist of several user interfaces. One user interface will be used to setup the scenario, and simulation parameters. Another interface will be used as a representation of what is going on at the crisis center. It will also contain a graphical representation of the area that will show the incoming reports of the people that are currently in the simulation area.

AI-component

This component will consist of a knowledge based system, that, based on decision rules that are derived from first-hand experience from experts and real-life complaints from people that observe thing in the environment, helps in deciding what probably is the most realistic scenario that is currently happening. This is of critical importance in the first stages of the development of a crisis, when not much information is known and a first hypothesis can be made through a knowledge-based system. Also the expert system can reason about actions that the crisis center should do, or, in case of a simulation, perform those suggested actions so the user can see what the effect of the action would be. To model the crisis management component in this system, we tried to incorporate the concepts of Distributed Decsision Making in MACSIM. The agents in the world can observe things and have their own conception of those observations, therefore the agents in the world have context awareness. The crisis center, besides having context-awareness based on incoming messages, also has the capability of making command descisions based on the incoming messages and a-priori knowledge.

Mobile Component

In the future, the crisis response components should also be able to be used in real life (no simulation) and for this, a mobile component should be created that can be run on a handheld, that enables the user to create reports based on an icon based application such as ISME [Sch05]. These reports are then treated in the same fashion as other reports, and can be used in the expert system in the same way to help determining what the most probable crisis situation might be.

IO Component

This component will take care of the regular file IO operations that might be necessary inside the system, such as loading and saving files. Examples could be the loading and saving of scripts, or saving graphical representations while running.

1.4 Approach

During the period of June 2005 until August 2006 the research and programming was conducted. In this section the route to the current system as it is right now is being described.

First a literature survey was being conducted to determine what kind of software tool for simulation of gas and fire incidents were available. From all the available models and tools, the ALOHA tool and the SAFER Realtime System were being chosen as important sources of inspiration. Nevertheless we chose to create our own software because of the fact that the software we used as a source of inspiration was not freeware (in the case of Safer Realtime) and was not open source (in the case of ALOHA and Realtime). Therefore the software development on top of these product became very difficult and own software development became a more attractive option.

Also a selection of people who could be interviewed was made. Then those people, coming from the most important crisis response agencies for our project, the Fire Department and the DCMR, were interviewed to acquire background and inside knowledge that only case specialists could provide us. The results of those interviews are being presented in Chapter 2.

These interviews were also used to conduct requirements analysis for the MACSIM project as a whole, which will not end with this thesis project. At this moment it was important to create a proof of concept, but when other people continue working on MACSIM, the focus will be more on creating a product that people in the field are waiting for. Therefore those interviews with the domain expters were extremely useful to get ideas for user interfaces and future functionality that can be designed and implemented in upcoming implementation iterations. At this point, due to time limitations, we did not yet have the time to get back to the domain experts to receive feedback on the requirements and user interface ideas for a more final version of MACSIM. The ideas for future interfaces are presented in Appendix A.

Before the actual design of the MACSIM system could start, it was important to figure out what features should be included that were already available in other tools. From this a list of features was created that is listed in the previous section and was used to start the preliminary design.

The actual multi-agent environment could not be implemented before there was a world model in which the simulation takes place. This world was designed and implemented in such a way that in later versions of MACSIM new features are added easily and to include more types of incidents. When the world was implemented, it was time to include a simulation environment that could handle the unfolding of a scenario, i.e. process the consequences of an event inside the world model during the course of the simulation time. Then it was decided that for future use it would be a good idea, for compatibility issues, to use the JADE environment as the multi-agent system middleware. This meant that the agents and the simulation components had to be designed in such a way that they ware compatible with the JADE agent standards.

When the agents were implemented, it was time to create a language in which the different components would communicate with each other. Because in the ontology by Fitrianie [FR06] a very wide scope of possible incidents was already available and could be converted quickly to JADE-compatible messages, we chose to integrate this ontology for the sending of messages. When the designed agents should receive messages they should be able to react. They should be able to execute agent actions. Those actions were programmed when the messages became available.

From then on the focus was on creating the AI-reasoning component of the system, which meant integrating a rule base inside the agent. Because of the easy integration with Java components (for instance the Java-based ontology from Siska Fitrianie) the Jess system was chosen. When the rule bases were inserted, the link between the agent actions in Java code and the activation and firing of the agent rules had to be made, so the agent could be able to react based on preliminary reasoning on the input on a certain time instance. Also the way in which a crisis was being assessed was defined here. To this point there had been no focus on the user interface. When the internal functionality was working, it was time to focus on the user interface to demonstrate what the system is doing in the background by means of a set of simple GUI's. For this purpose we made use of Tatomir's 3MNews software [Tat06] that was already being programmed earlier on to process messages based on Fitrianie's ontology as well. The components used from 3MNews enable us to generate a picture with a map on which icons can be placed that represent messages that are coming in. Based on these components, the system was debugged and tested to enable testing of the scenario's described in Chapter 6. Based on this testing, suggestions for further development and more elaborate testing were made. They can be found in Chapter 7.

1.5 Thesis Overview

In Chapter 2, the underlying theoretical background and related work that was the foundation for the design and implementation of MACSIM is discussed. Some projects referred to in this introduction will be discussed in more detail in this chapter. The global design of the system is given in chapter 3, in which the fundamental concepts of the system are being introduced. For a more detailed software design and algorithm-based look on the system, see chapter 4, called Detailed Design. In the next chapter Implementation, the results of the efforts until august 2006 are presented, in an attempt to give an overview of what has been implemented (and in which way) and what still needs to be done. Chapter 6 presents an evaluation of the current system based on two predefined scenarios that are being tested inside the system. This helped to determine what are the strong and the weak points of the current system implementation. Finally in Chapter 7 we present the conclusions and recommendations for future research efforts related to MACSIM.

In Appendix A some propositions for user interfaces are added, based on the requirements analysis interviews with the domain experts. These interfaces are just prototypes that are not yet implemented, but will be the source of inspiration for User Interface development during future implementation increments. Appendix B contains the transcripts of the interviews held for knowledge acquisition and requirements analysis. Because the interviews were held in Dutch, the transcripts will also be presented in Dutch. In the case of the interview with Sjaak Seen, we include the abstract of his thesis as a transcript because most subjects he addressed during the interview were mentioned very clearly in this thesis. A short explanation about how to edit MACSIM data structures is mentioned is given in Appendix C. The paper written for the CGAMES '06 conference in Dublin is added in Appendix D.

Chapter 2

Background Theory

In this chapter we will discuss the theoretical background of the research conducted in this thesis project. We will give an overview of the sources and background that were a key inspiration to develop MACSIM as it is in its current form. In the first section we will focus on crisis management itself, how is it being performed in the Netherlands and what kind of research has been recently done. In the next section we will focus on the related work on which the different components that had to be implemented are based. In this section we will describe the different components that are included in MACSIM and what earlier performed research and projects are related to it.

2.1 Crisis Management

2.1.1 Crisis Management in the Netherlands

This section is dedicated to giving a description of the way crisis management and crisis response are currently organized in the Netherlands. Most of the material in this section is derived from interviews taken in the period of august 2005 to April 2006 with Rene Kuijper, chief crisis center of the Rijnmond environmental agency (DCMR), Richard van Haagen, chemical expert of DCMR, Leo Kooijman, former chief of crisis management of the Rotterdam Fire Department and Sjaak Seen, deputy chief of the Southern Rotterdam Fire Department. Mr. Seen's master's thesis was a very useful source of information as well. Furthermore a lot of information was taken from the inaugural speech from September 2003 of Prof. Ben Ale, associated professor at TU Delft for the chair of Safety management.

First we will give a bird's eye view of the history of crisis management in the Netherlands after the Second World War. The reason for this is that it gives an idea why crisis management is currently organized as it is right now. Then a description of all the parties involved is given. This list of parties will of course not be an exhaustive list, but the ones that are most important for the project will be discussed. In the next section we try to show which protocols and parties come into play in a crisis situation. Also the different stages of a crisis situation are discussed. Furthermore we discuss the problems and pitfalls that exist in the current situation, as experienced by the people in the field. Finally we discuss which IT innovations the people in the field are already implementing and what their experiences with these products are.

Short History

In 1945, after the end of the Second World War, the Dutch government decided to join NATO to be more prepared for an attack on the free countries in Europe. In the case that the NATO treaty was activated, it could imply that the Dutch army should be conducting military operations on foreign soil, thus leaving the Dutch population unprotected. In those days the military was in charge of crisis management. So

as a replacement of the regular army, another semi-military organization was formed, that was idle during peace-time and whenever necessary it was activated to perform duties like providing shelter to the people during an air raid, transport of strategical goods like gasoline etc. This organization was called Peoples Protection (BB in Dutch).

The BB was a static organization that was difficult to mobilize because in peacetime it wasn't present. And as the Cold War in Europe progressed, the chances of a military attack on European soil decreased so the actual job of BB was becoming more and more obsolete. Their main function was to help the people in case of an attack, but they weren't really prepared for large-scale crises or disasters. The main reason was that it took time to mobilize the entire organization. This time is available in wartime but not in the case of a civil crisis or disaster, in which the activation of the crisis management team should be completed in a matter of minutes.

After two large disasters in the mid-seventies, the Dutch government decided to abolish the BB and to transfer most of its tasks to the fire department, the police, local hospitals and the Red Cross. The main coordinator in case of civil disasters became the fire department. The way in which crises were managed stayed mostly the same, the only difference was that it was now executed by a permanently active organization. This gave some problems in practice, because the crisis management strategy still assumed that there was some sort of enemy and that command structures were still very hierarchical. A lower layer of command could not do something without explicit permission from a higher rank. In situations where there should be acted in haste, this is not a desirable situation.

In the mid 90's after the Bijlmer-disaster, in which an El-Al Boeing crashed into two residential flats in Amsterdam, it was decided that there should be more efficiency in crisis management efforts to reduce the amount of communicational overhead. Therefore initiatives were taken to enhance the communicational features of the agencies involved. One of them was C2000 (we will discuss this in more detail later on).

For the near future and with the disasters in Enschede and 9/11 still fresh in memory, the concerned agencies are still looking for new solutions to enhance their preparedness for disasters. It is for this reason that the agencies are looking for performance enhancements in IT solutions [SLOdS05].

Description of the Involved Parties

First we give an overview of the different parties involved, then we give the names of the discussion structures in which those people come together in crisis management and response situations.

DCMR (Dienst Centraal Milieubeheer Rijnmond)

The agency for environmental quality in the Rotterdam/Rijmond area is called DCMR. This agency is responsible for monitoring the air quality in the Rijnmond area. It is dedicated to the Rijnmond area alone because in that area there is a huge concentration of petrochemical industry that justifies a special local organization. For other parts of the Netherlands, the RIVM (Government association for Public Health and Environment) is in charge.

One of the main tasks of DCMR is registration of complaints; all complaints about incidents that could be an environmental threat are being reported to the DCMR call center. This could be warnings of companies that they are doing something that could cause complaints in the near future, but of course also calls from concerned members of the general public from the Rijnmond area. Small emergencies are mostly reported to 112, the national alarm number, for larger incidents most people in the Rijnmond area call the DCMR.

The second task of DCMR is assessment. In cases of an emergency the call center becomes a crisis center and from then on the crisis center tries to clarify in a short timeframe what is the cause of the

complaints that are coming in. They try to create a hypothesis of the possible crisis situation based on the data that is available at the crisis center. Also they can initiate an exploration mission by chemical experts from DCMR to check out what is going on. In more severe cases, a complete gas measurement plan is being executed that for a larger area gives concentrations of gases. For this, DCMR has their own analytical equipment. It mostly concerns taking air and water samples on the (presumed) crisis location. In the case of extreme emergencies only firemen or chemical experts from DCMR are allowed near the crisis area in special protective clothing, and the area is completely sealed off. When the gas measurement plan is initiated, it requires several measurement crews to measure at several predefined measurement locations to accurately determine the dispersion of a certain gas. The contaminated area is then being indicated on top of a map.

The third task of the DCMR is giving information. The crisis center informs members of the general public during a crisis and does this as accurate as possible based on the information that is available at the crisis center on that point. In addition to that, the police informs people with cars with speakers on top, with which they can drive through areas of concern. The crisis center also informs other services that are involved in the crisis management effort, such as the Fire department, police and for instance government officials.

The DCMR crisis center functions within the official crisis plan that has been acknowledged in the Rijnmond area. It serves as a support service for the crisis instances that will be discussed later on in this chapter, the COPI and the ROT. Fortunately, the Rijnmond area has never been hit by a crisis that was so severe that every aspect of the crisis plan had to be executed. On the other hand, on a regular basis there are crisis drills. In practice, however, they have never had a real situation that was so extreme that it causes a large wave of panic across the general public, like for instance the period that preceded the Katrina hurricane in New Orleans. Therefore all incidents that they had to cope with so far developed very orderly [KBR05b].

DCMR Chemical Expert

A special actor from DCMR is the chemical expert. In the case of an emergency he is appointed Regional Officer Dangerous Substances (in Dutch : ROGS). In this function the ROGS gives detailed information about chemical substance and their behavior to the police, the fire department, medical officials and the government. This department from DCMR falls under the command of the fire department in crisis situations. They have their own protective clothing, and because of this they are capable of conducting measurements in very dangerous circumstances. These are the only experts from DCMR that are directly involved in crisis response in the case of a terrorist attack. Officially DCMR is only in charge of pollution and environmental issues, so a terror attack for most of DCMR is out of its scope.

Most of the time, the job of an ROGS is conducting diplomacy between the other agencies involved. In the case of conflicts of interest the ROGS is a mediator. Official the ROGS is only an external expert, but he has a lot of authority. If the ROGS says something it's almost impossible to ignore it or do something else because afterwards an official will be held accountable for ignoring an advice from an ROGS.

As a chemical expert it's very important to have a knowledge network. This contains information resources and also people from other organizations that might know more about a particular subject than the ROGS. If the ROGS doesn't know something, he should be able to figure out where to find it as fast as he can [vHBR06].

Fire Department

The fire department is of course mainly concerned with fires, but in crisis situations they have a lot more jobs to attend to. Currently, for practical reasons, the fire departments are divided in roughly the same areas as the police departments. This makes their job as coordinator in case of crises a lot easier. In case of a crisis, the most important person on the crisis site is the chief fire officer (in Dutch: Officier

van Dienst or OvD). He is in charge of the crisis rescue effort; he decides whether the alarm sirens in the direct area should be activated. Also the official gas measurement plan that can be activated is being done so under his command [KBR05a].

Local Government

The local government basically consists of the mayor, and in crisis situations that have a larger scale of impact, the mayors of more communities. The mayor is in charge and politically responsible for the crisis effort from the moment that the crisis can become a threat to a large community. He might not know anything about technically managing that particular crisis, but he is the one that has to take difficult decisions that affect the lives of the people that live in a particular area [KBR05b].

Police

In regular crises, the police gives support to the crisis effort, like for instance keeping the general public on a safe distance or providing information through cars with sirens on top that drive through the area of concern. In other cases however, when there is a situation that could be a terrorist attack, the police becomes coordinator of the crisis effort. Even in the case of a biological or chemical attack, the police is in charge of the crisis response effort, and all other agencies have to execute orders from the police. Of course, the fire department has an important say in these cases but the final responsibility lies with the police [KBR05b]. Also, if a regular incident might have a criminal cause, the police might want to seal off the area to conduct an investigation. In some cases this might give problems during the actual crisis effort. In section 2.1.1, this will be explained in more detail.

Medical Official

The medical officer is in charge of giving medical advice in case of a crisis. Most of the time the medical official is the head of the community medical agency or GGD. He is also in charge of giving vaccinations to the general public if necessary. Also the medical officer gives information to the general public about the medical risks during an incident.

Communicational Structures

The crisis management and response system in the Netherlands knows 3 official structures of communication that can be used in different levels of emergency. They are discussed below [SLOdS05], [KBR05b].

COPI (Dutch Abbreviation for Command Structure at the Scene of the Incident)

In the case of a small incident only a COPI is being installed. They are located inside a red container a couple of 100 meters away from the scene of the incident (of course when the situation does not allow such a distance it is moved further away). All operational agencies and officials from a company where an incident is happening are present. The main function of a COPI is to do everything that is directly necessary for resolving the incident.

The advantage of a COPI is that it's multidisciplinary and it works as a team. Each decision that is being taken is taken as a decision that is supported by the entire COPI. This increases the sense of responsibility for the individual team members, and the sense that there must be thought in favor of the common good, instead of the good of the individual team member.

One of the main problems of a COPI is that it requires a lot of teamwork. This implies a lot of practice, and indeed that's something that needs to be done quite regularly. But a great advantage of COPI is that because all decisions are being made as a team, there is shared responsibility and therefore there is a much stronger case against lawsuits. Mostly because the decision has been made as a group and it was the best alternative possible that has been chosen that served the common good.

ROT (Regional Operational Team)

The ROT is an addition to the COPI. If an incident takes very long or has a large area of impact, then the people at the COPI need to be released at a certain time. Also the firemen and police officers should be changed. Their main goal is to take care of practical issues concerning the crisis response effort itself. Also they plan large-scale operations that are ordered by the COPI, such as a large-scale evacuation.

GVS (Dutch for Local Security Staff)

The GVS is the political decision structure that comes in to play in case of a large-scale incident. This organization contains the public prosecutor, the Head of police and the mayor, also the medical official and the chief fire commander. The GVS is also responsible for the external communication. They issue official statements for the general public. This organization is only installed if it's a very large-scale incident in which an entire region is involved.

Despite the fact that the chief fire commander is operationally responsible, the mayor is politically responsible and he should make all management decisions. That can be a problem because he might not be a subject expert, but it can also have advantages because it means that the chief fire officer can concentrate on solving the crisis and he can "outsource" difficult political decisions that have pros and cons on each side to the mayor, so any political problems afterwards will be the mayor's problem.

The Crisis Situation: Training and Reality

Training

Because incidents can happen at any time, it is of vital importance that all agencies involved in crisis response practice as much as they can as realistically as they can. On different levels several kinds of training are being conducted. The fire agencies have their regular fire drills, but also the COPI environment is being trained regularly. Mainly because in a crisis event it is the first installed level of communication. In some cases more levels of command are training at the same time. This kind of training is called integrated crisis training. [KBR05a].

Most of the times training consists of putting persons in a real life setting (e.g. a COPI-container) and let them work things out in a scenario that has been planned in advance. Those scenarios are prepared very thoroughly. The procedures that for instance firemen or COPI members have to follow during crisis situations are often quite generically applicable. They are trained in such a way that they have to do the same every time whenever possible. The reason for this is that at least it is trainable and the crisis responders at least know something to do in case of a crisis. To start thinking about alternatives is not an option when it's almost too late. The bottom line is: train people in such a way that they only do something different than what they have learnt if and only if they are sure that it's useful and no other trainable tasks have priority [KBR05a].

Scenario-based COPI training usually lasts two days and is being executed 18 times a year at different locations in the Rijnmond area. When such training takes place at a company they usually have three containers installed in which 3 COPI training session are going on concurrently. In those two days 3 scenarios are being executed. People from the fire department or from the DCMR are responsible for the making of the scenarios and creating the data. The exercise participants have to cope with this data. Most of the times a COPI training consists of a book with the messages that arrive at a certain point in time. From time $t = 0$ the scenario is unfolded and it is a simulation role-playing game.

The type of messages that come in at a COPI training can vary. Some of them are essential and cannot be missed, other ones are just plain nonsense and it's up to the participants to figure out which message belongs to which category. The messages can come from the people you are commanding at the COPI but also from your own superiors. Especially the latter category can be very difficult for people because

they feel honored that they are for instance talking to the boss of the plant or another high-ranked person, but what they should be dong is say to somebody that they don't have time for them at the present moment, because there is a crisis to be solved [vHBR06].

First the participants see a movie about what has preceded the simulation. And then the scenario begins. It is very difficult to organize a scenario like this because it not only implies that the participant should be able to adapt to a changing situation, but also the director of the simulation should. The participants can always come up with a point of view that the writer of the scenario didn't think of in advance. For this they try to find a balance between fixed scenarios and scenarios in which there is a certain level of flexibility, so the director of the training can intervene if necessary.

In integral crisis training, like for instance the Bonfire training [Rem05], [COT05], which was a full scale crisis drill in which everyone up to the minister of internal affairs was taking part in the simulation. This immediately caused problems because of the number of hierarchical levels that were participating in the same scenario. It is difficult enough to keep a scenario running that you planned for a COPI, let alone a scenario that takes care of every single crisis manager in the country. This requires very rigid planning and no deviations from the scenario whatsoever [KBR05a].

Reality

Everyone involved in the crisis response system has their own responsibility for the execution of tasks, as long as their superiors are not overruling them. To a certain extent their safety is their own responsibility. If someone has the idea that his personal safety may be at stake, they have specific instructions to take care of their own safety first before executing their job.

In the event of a crisis members of the police, firemen and responsible authorities meet in the COPI. In the COPI all orders are given to the people in the field who are busy with the rescue operation. In case of an incident that takes long, a ROT is being added. For large incidents, the GVS is installed. In emergencies, the public transport companies are being deployed for evacuation. This only happens, though, when there is enough time to do so. Otherwise the regular directive is followed: to stay inside with the windows closed. It is by far the best alternative [KBR05b].

Before that, officially the following should have happened: an employee at a company that notices an incident is obliged to report this immediately to his superior if he thinks the incident can be hazardous to the environment or the community. The company then immediately calls DCMR with an incident code. All chemicals have a code so everybody knows which chemicals are being used. After that, a telephone conversation takes place, in which the fire department DCMR, and the police are involved. A series of questions about everything you might want to know about the incident is run through. After the list there is asked if anyone has any more questions. After that the COPI is being set up [KBR05b].

In the situation that there are reports from the general public, the DCMR will start measuring and checking out the situation if there are only complaints. If it's dangerous the fire department will check it out. DCMR has two tasks in these kinds of situations: ROGS and leader of the gas measurement plan. The leader of the gas measurement plan is working under command of the chief fire officer. The chief fire officer can initiate the gas measurement plan if there is a chance that the health of the general public may be in danger. The gas measurement plan leader has DCMR crew and equipment (measurement devices) and measurement crew from the fire department at his disposal. In total this means a measurement crew of approximately 70 people [vHBR06].

The most difficult complaint to determine is a boat that, while moving is illegally dumping chemicals. The complaints are often coming from another direction every time and when people start measuring at the indicated location there is no source to be found. Only much later, when the multiple possible locations are being put together, they can track it back to a boat [KBR05b].

When it's necessary for the ROT to meet, in the Rijnmond area they come together at the World Port Center in Rotterdam harbor. There they have all communicational facilities at their disposal. The only thing that they explicitly do not have in the ROT is a TV screen. They don't want to be disturbed by the messages that are being broadcast by the TV stations. It could be the case for instance that the TV station is showing a plume of smoke in the screen because it's easy to see and to show to the TV audience. But in this situation the danger could be somewhere entirely different, but not visible for a camera. The plume of smoke could be a necessary measure to prevent another problem, for instance. To prevent that the attention of the ROT gets focused on the plume of smoke instead of something that is the real problem, but not visible to a camera, the people at the ROT prefer not to have live TV images at their disposal. Besides that, television crews nowadays often are earlier at the location of an incident that the actual rescue workers, simply because at the first message they get (from for instance scanning the police radio) they are on their way, to be the first to be there because of competition with the other TV networks [KBR05b].

At the GVS (which is also located in the world port center), which is in charge of external communication, they are interested in TV- images because those TV images are the things that questions are being asked about. So they do have a TV-screen at their disposal. This example illustrates that even in the same building the perception of a crisis can be entirely different between different parts of the crisis management effort [KBR05b].

Therefore it is not too strange that the GVS and the ROT are at different rooms in the same building, because otherwise these different perspectives can get mixed up. For all information input coming in at one of the parts of the crisis management effort the question should always be whether the information is necessary for that particular part to do the job right, otherwise it will be considered obsolete.

Pitfalls and Problems

In this section some problems that can come up in every day practice in the current situation are discussed. These examples were given by the people that were interviewed, so any account or complaint mentioned here is based on the experience of that interviewee. It does not represent the official opinion of the organization that he works for, and is only an illustration of what is wrong about the current situation.

At the crisis management section of the fire department they prefer to have dynamic disaster management plans, but the problem is that if one makes a rule for every possible event that might happen, those disaster management plans become huge. A crisis manager should know at each moment what he should do, so he doesn't really have time to browse through hundreds of pages with disaster management procedures. And most of the times in a crisis situation no decision at all is worse than a bad (read: non-optimal) decision. They tend to see a disaster as a complex mixture of events that all happen at the same time and as long as they decompose this huge chunk into tiny mini-events that rescue workers do know how to handle. In this way the rescue workers can concentrate on what they are good at and don't have to do too much thinking on the spot [KBR05a].

On the other hand, Richard van Haagen of DCMR advocates that the lack of flexibility and rigidness in crisis management as a whole bothers him. In his opinion the western world and in this case crisis management in specific is obsessed by putting everything in rules and procedures [vHBR06]. Especially in the political and management departments people think that if they have procedures on paper, everything will work out fine, and they don't even know what they are talking about. This seems unfair, because actual crisis response work is being paid less than desk jobs. For chemical experts, entirely to the contrary of the fire department's motto, it is important that procedures are not fixed things. They are a starting point but there is no good reason why there can't be a deviation of a procedure if the situation asks for it. They are much more flexible with procedures than the people from the fire department. It

should always be an option to deviate from the procedures.

Something that is currently not a regular practice in the COPI training is the situation in which the COPI becomes a victim of the crisis itself. This could be either the case in a toxic or nuclear incident, but also in the case that a COPI becomes the victim of a terrorist attack. What should be done is putting another COPI around the victimized COPI. In the Netherlands this is still a novelty. In countries where they have a lot of experience with terrorism, like Northern Ireland or Spain, this is very common practice [vHBR06].

Another problem during crisis management and response is the perception from the general public, which chemicals are dangerous and which ones are not. Most people think that LPG transports are the most dangerous transports out there on the road. Indeed, it can be very dangerous and the damage afterwards can be devastating, but the reason people think it's dangerous is based on the fact that those transport take place very often. SO_2 transports are potentially much more dangerous but they are only carried out twice a year. But the results of an explosion of an SO_2 tank can be devastating, with thousands of casualties [vHBR06].

Also the general public thinks that all gases that have a sulfuric odor are dangerous. Consequentially the general public only complaints about that gas, and another gas that might be much more dangerous might be undiscovered. The only reason for this is that it doesn't have a smell like rotten eggs [vHBR06].

During crisis response there are some cases in which the fire department would have liked to clean up the mess afterwards after an incident but they aren't allowed to. This is because of liability claims and legal action from the government. They first want to investigate whether the company where the incident took place is criminally liable and insurance companies have to pay for the damage. In certain cases this might give a problem because the chemicals that were released have a very poignant odor, and the longer the situation stays uncleaned the longer the direct environment has to put up with it [KBR05b].

At the ROT they don't mind, but at the COPI the lack of valid and helpful TV-imagery is considered a problem. On a lot of public places nowadays there are CCTV cameras monitoring every move we make, but as Murphy's law states, those are obviously not monitoring that particular area that the COPI is interested in. That's why at the COPI or DCMR they are very grateful to people in the area with digital cameras that send images via e-mail to the DCMR. Also they have installed webcams on places where there are no CCTV cameras available [KBR05b].

The ROT convenes in the World Port Center, in the middle of the Rotterdam Harbor. The decision if the siren will be activated is taken here. If the siren is activated, then the general public should tune in to the local emergency broadcasting station. In the Rijnmond area this is Radio Rijnmond, which means that they broadcast all official statements by the government, most of the time these messages come directly from the GVS. The problem is that Radio Rijnmond is also an independent news station. This means that besides giving official government statements they can also go out and explore on their own. This gives them a double role, the role of news maker and official broadcaster of government messages. This can give problems for the general public to distinguish official certified news from speculation or independent exploration by the station itself [KBR05a].

Officially, TV is still not considered an official disaster media in the Netherlands, the official directive in disaster management plans for the general public is still whenever the siren is activated, to go inside, close all windows and doors, and have a battery powered radio tuned in on your local emergency broadcasting station. This is because radio contact is very easy to maintain, even when power is cut off [KBR05a].

Sometimes because of traffic jams it's very difficult to get everyone in the right place. Naturally all emergencies come unannounced so officials that need to be in the ROT or COPI might have to come

from everywhere. This sometimes calls for unorthodox measures. In some cases it is more practical to move officials and experts around by bike or boat than by car. When they are transported by car, executives don't get flashlights on their cars. Experts from the DCMR do have a flashlight on their car [KBR05b].

Innovations

In this section a couple of examples of IT-innovations are being described that are already implemented in the recent years. Also a couple of suggestions from the experts in the field can be interesting starting points for new innovations.

Multi-Team

From the fire department there has been the initiative to develop the system Multi-Team. This system tries to streamline the communication between different crisis response agencies. The system is based on Microsoft Internet Explorer, but it doesn't use Internet for its data traffic. This has been a conscious decision because Multi-Team should always be online and that, unfortunately, with the Internet cannot be guaranteed.

Currently Multi-Team uses a special secured line from KPN, but in the near future they will switch to a dedicated robust crisis response communication system, called OOV-net (OOV-net stands for Public order and safety in Dutch). This is a very secure network connection dedicated to providing the communication layer for all government-based public safety applications. All tools the government uses for public safety will switch to OOV-net in the near future [KBR05b].

The key strategy to make Multi-Team work is proper authorization. Multi-Team is a system that allows intelligent combining of databases of different government agencies during a crisis situation. It is not desirable that other people read that kind of information when it's really none of their business. Only the right people sould be reading that sort of thing and then only if it's in the case of an emergency. Currently they are working hard to make those authorization policies work [KBR05a].

The only problem with Multi-Team is that every time something has been developed, the project management feels that there is so much more to add to the system. The continuous question is where the system is going to end. What was unaffordable in the planning stages of the project is affordable at the implementation phase. For instance, in 1998 the addition of a complete GIS system was out of the question, now every car has built-in navigation software and it can be easily added in a new version of Multi-Team.

As an addition to Multi-Team currently experiments are going on with adding a data-terminal on a fire engine. In this way the results of research in the crisis center can become available on the fire engine while they are on their way, but also dynamic routing of the fire engine is possible. Because of the traffic, it becomes essential to find the correct route to get to an incident fast. The fire engine should be able to arrive somewhere in a couple of minutes.

Multi-Team has been developed by TNO Defense and Safety, it has been used in a COPI environment during training and real life situations. This has learnt them that there is a lot of information available inside Multi-Team, but it's up to the user to combine it in the right way. Even information that seems trivial to one person can be essential to another, so distribution of information is also essential. Currently the Multi-Team project management is busy writing special disaster scenarios to test the complete functionality of Multi-Team [KBR05a].

C2000

C2000 is the communication system that is used between all government agencies and crisis response agencies such as police, fire department, ambulance and government officials. C2000 is an advanced

version of the GSM for crisis responders in the field. It is capable of having conversations on a one-on-one basis, but you can also have group conversations. After a delay in release of almost six years, the preliminary test results are optimistic, but there are also some critical notes.

The most critical one is that communication protocols should be adapted to the new system. With the current protocols C2000 allows too much chatter on the line. Currently efforts are being made to adapt the communication protocols in such a way that conversations develop in a more orderly fashion [KBR05a].

At DCMR the chemical experts have to use C2000 as well but they are a bit more skeptical about it. They believe that in some situations the regular mobile phone is much more practical. They regularly use the ordinary cell phone as a backup. ICT innovation is nice, but nevertheless the regular systems should not be ignored as a backup. For this DCMR Experts, besides their electronic equipment always have reference books at their disposal. They consider electronic devices as potentially unreliable [vHBR06].

Integration of New Media

At the DCMR call center, they have started to use web cameras on places that the regular CCTV misses. This is done because the COPI can find some visual information quite useful. Also digital images from people that are sent by e-mail are currently used at the call center and at the COPI for additional visual information to assess the nature of the incident [KBR05b].

Additional Suggestions for IT Innovation

DCMR considers a simulator in which simulations can be run which can realistically model the overflow of information that a crisis response team has to cope with. Such a system should also be able to help in filtering useful information out of useless information. Another good feature would be if the discrepancies between different levels of crisis management could be shown during a simulation while a crisis situation is going on. It could happen that at the GVS they are panicking and call for the siren to be activated, while the DCMR experts do not understand what the fuss is all about. In any way, this system should have the opportunity to let agents work outside of regular procedures, just like the DCMR experts are used to [vHBR06].

At the TU Delft, Prof. Ben Ale in 2003 already advocated the use of simulation and computer games technology for aiding decision makers and rescue workers in their training abilities. He thinks that the most important factor is the way you can represent the disaster in such a way that it serves training facilities best and it also makes sure that it is very realistic [Ale03].

The fire department is enthusiastic about Multi-Team, and wants further research into the possibilities in which the existing system can be expanded. Also they want to look into the possibility that virtual conferencing may offer for creating Virtual COPI's and virtual officials, that are part of the COPI structure but do not necessarily have to be on the right place, which could for instance significantly improve setup times for a COPI [KBR05a].

The thesis of Sjaak Seen furthermore stresses the need for the addition of an information manager on all levels of communication (COPI, ROT and GVS) to filter the right type of information for every one who might need it. A filtering mechanism, as suggested by the DCMR would in this case be an extremely practical tool and a very interesting research object [SLOdS05], [SBR06].

2.1.2 Crisis Management Research Examples

In the recent years, there was a significant increase in the amount of research that has been conducted in the field of Safety management. In this section we try to give some interesting examples of research on crisis management and we indicate if and how the particular conclusions drawn can be applied to the

design rationale of MACSIM or crisis management tools in general.

The Enschede Disaster - May 13th 2000

In the Dutch town of Enschede, in the middle of a residential area, a Fireworks company called SE Fireworks was located. It made fireworks for the entertainment industry and had on its facilities a lot of containers in which large quantities of fireworks were stored. What exactly happened on the fatal Saturday that was the cause of a fire on the premises of SE Fireworks is still subject to a lot of discussion, but the fact was that on 15:03 the fire department of Enschede received a report of a small fire on the site of SE Fireworks. When the firemen arrived, they started to extinguish the fire as far as it was visible.

Somewhere around 15:18 on the terrain fireworks started to go off, mostly light fireworks. At this point the firemen try to do the best they can to get the fire under control, until the fire spreads to a series of containers, which apparently contain very heavy fireworks. At 15:32 the series of containers explodes, resulting in the largest fireworks disaster the Netherlands has ever seen, an explosion in intensity somewhat equivalent to a Second World War bomb. The consequences were devastating. An entire residential area was demolished, there were 22 casualties (4 of them firemen), and 950 wounded. The total damage has been estimated on approximately 500 million euros [OdBE$^+$01].

The initial response to the fire was adequate, but according to the actions of the firemen that were at the location they were not prepared for the level of fireworks that was stored in there. They would probably have chosen to leave the premises as soon as they could had they know about the exact content of the storage containers on the SE Fireworks facilities.

On May 26 2000 a crisis research commission was installed, which on February 28th 2001 released their report on the fatal events on the day itself but also on the reasons why there could have been so much heavy fireworks in the middle of a residential area. The most important conclusion was that the Dutch government had been too lax towards checking the permits of the SE Fireworks facilities and that there had been much more heavy fireworks on the site than was allowed by the permits [OdBE$^+$01].

From all this it can be concluded that if the fire fighters were already informed about the level of fireworks in advance, they could have been able to choose another strategy to fight the fire, and in this way they might have been able to save the lives of innocent bystanders and their colleagues. Of course, if the government does not check and update the permits regularly, the fire department can never have updated information. If this information had been available, an information system that is able to share and combine multidisciplinary information while the firemen are on their way to the location of the crisis such as Multi-Team [KBR05a] could have been a practical addition to the rescue effort. Therefore a way to intelligently combine crisis information before arrival at the site could make the crisis effort operation more efficient and straightforward.

The Volendam Fire - January 1st 2001

In Volendam, in a small overcrowded café, the highest of 3 cafes on different floors in one building, just after midnight, a fire caused by light fireworks breaks out. Because of dry pine-wood on the ceiling with plastic decorations attached to it, in just one minute the fire spreads through the café and causes extreme heat. This, combined with the lack of a decent emergency exit, causes a huge level of panic amongst the visitors of the café, who desperately try to leave the premises as soon as they can through everything that looks remotely like an exit. This is difficult enough because of the huge development of smoke caused by the short but intense fire. When eventually the café visitors left the premises, on the streets in the area of the café there was no medical staff yet, so first aid efforts were conducted in a very haphazard fashion. Eventually 9 people died, and approximately 200 people were hospitalized, mainly because of severe burnings and breathing difficulties [Ald02].

The actual fire is over pretty quickly afterwards and therefore does not spread to the café's on the other floors. But because of the chaos everybody in the building leaves rapidly, causing a large commotion on the streets surrounding the disaster area. Some estimates are 1500 people are on the streets that moment. Therefore it's very difficult for emergency services to arrive at the disaster location. Police and fire department and ambulances were informed 40 minutes after the fire started and in a couple of minutes they are at the spot, but the problem is that because of the overcrowded streets the location is in a state of utter chaos. Therefore a lot of time is wasted with getting some sort of idea of what is going on, severely slowing down the medical treatment and checking what the status of the fire is at the disaster location. As was concluded in the report concerning the disaster, the overcrowded streets combined with communicational led to a situation in which the crisis management effort was slowed down by at least 2 hours [Ald02].

In the investigation afterwards it is concluded that according to the building permits there were officially allowed 84 people inside the café concerned, and on that particular night there were at least 300 people inside. Besides that, there also wasn't a decent fire-exit at the café that should have been there according to regulations. But, as in the case of the Enschede disaster, a lax government let this happen.

In this case the initial crisis response by the respective agencies was good, the situation was investigated and correctly classified by the authorities and they were on the spot very fast. But from then on things started to go wrong. Mainly because of the lack of communication between the services, but also because of the large crowd of people that by then has gathered on the location of the incident. So this time not the interpretation of the incident but the communication was a structural problem. Therefore, relating this to the design rationale of a crisis management simulator would mean that it is very important that inside a crisis management effort the communication in severe crises can be trained, and that the level of chaos can be realistically modeled. This is also suggested by Rothkrantz [Rot06]. Also the current implementation of one single communication device (C2000, [vHBR06] and [KBR05a]) could also have been helpful. C2000 was not yet available at the time of the Volendam disaster as it has only been released in April 2006.

The 9/11 Terrorist Attacks - September 11th 2001

The event that had so much impact on the western world, that it does not need too much introduction. Three airplanes, hijacked by terrorists fly into the two buildings of the World Trade Center in New York City and the Pentagon in Washington DC. A fourth hijacked plane crashed south of Pittsburg, Pennsylvania, either planning to land onto the White House or a nearby nuclear facility. The results are also well known: a massive fear of terrorism across the world and more than 2100 deaths, and the destruction of one of the most prestigious landmarks of the New York City skyline.

Since it is one of the key events of our time, of course there have been masses of research efforts on the rescue effort and the intelligence agencies' lack of information to prevent such an attack. Because it is beyond of the scope of this thesis to go into minute details concerning the vast amount of sources that are available. We just want to point out the most interesting facts to be found in the Official 9/11 commission report [KHBV$^+$04] concerning crisis management and response as far as it concerns suggestions for technological improvement.

The first noticeable fact was that after the first tower had collapsed, there was no more mobile telephone traffic possible with the people that are were still inside the second building. Observation helicopters that tried to maintain contact with the people inside the second building were now cut off. This means that with lack of proper communication, the rescue workers were left to their own devices. If they could have been reached, then it might have been possible to inform the rescue workers (fire department and police) about the destruction of the other tower and give the orders to leave as soon as possible. Suggestions for information technology concerning this fact could for instance be found in the research

thesis of Paul Klapwijk [Kla05], which discusses wireless ad-hoc communication networks, that would enable communication devices, in a lack of a central server, to change to wireless networking structure that maintains contact between devices through contacting other devices that are in the vicinity.

The other noticeable fact was that there were problems in the communication between the policemen and the firemen in the two buildings. There existed a conflict situation between the police units and the firemen about who was to give orders to whom [KHBV+04]. This situation is a typical situation that with intensive training in extreme circumstances provided by high quality simulation could have been trained more thoroughly.

The Space Shuttle Columbia Accident - February 1st, 2003

What happened in the case of the ill-fated final Space Shuttle Columbia mission was that on the launch of the Space Shuttle Columbia on its final mission, STS-107, on January 16th, pieces of foam of the booster rocket broke off and hit the left wing of the space shuttle. NASA engineers noticed this after two days, when launching footage on high resolution had been developed. Because it had happened before during earlier missions without any serious damage, NASA management was skeptical about suggestions by its engineers that the left wing might be damaged because of the foam hitting the left wing of the shuttle during launch. NASA staff ignored several requests of further investigation by NASA engineers during the following period, mainly because they believed that if there was any damage in the first place, then at that point in time nothing could have been done about it anyway [Geh03].

During the post-disaster investigation took place by the Columbia Accident Investigation Board, it was concluded that if NASA would have been able to find structural damage to the left wing within the first 5 days of the mission, it would have still been possible to conduct a repair or rescue mission. Those options were considered risky during the post-disaster investigation, but not completely impossible, like NASA management believed it to be [Geh03].

On the fatal attempt of reentry into the Earth's atmosphere, on February 1st, there were already very early signs that something was not right. A sensor on the left wing of the space shuttle read abnormally high temperature values very early on during the reentry procedure. These values were recorded and found afterwards, but not detected or interpreted during the crucial moments. This appeared to have been a key clue for what was going to happen in the next minutes. Because of the extreme heat and atmospheric gases coming in through the damaged left wing, the structure of the space shuttle began to destabilize, causing the Space Shuttle to become uncontrollable, and eventually disintegrate. Among the 7 crewmembers there were no survivors [Geh03].

This example shows that every bit of information that is there should be interpreted as equally important. The problem is that there is so much information out there; human beings can interpret not all of the information. Therefore expert systems that maintain regular contact with real human monitors are a preferable option. If NASA engineers had observed those sensors on the left wing, the existence of a crisis situation might have been foreseen before reentry of the atmosphere. It is not sure whether at that point the crew of Columbia could have still been saved by then, but at least the recognition of the crisis itself would have taken place much earlier.

With this example, though not directly related to everyday crisis management, we attempted to show the importance of intelligently combining all available sensory information with the help of intelligent software based expert systems. To a certain extent this conclusion will be incorporated inside MACSIM's design rationale. In MACSIM, the combination of reports of smart sensors and observations made by the General public are a first set of clues to the expert system of what is going on.

Experiments with Adding a Camera on Firemen Equipment

It is an interesting development that because of the rising interest in crisis response research a lot of IT solutions are developed that might be able to help rescue workers during the execution of their duty. But the central question in these efforts should always be whether the everyday routine of rescue workers, who have to make decisions in split seconds, is not too much disrupted, and whether the rescue workers in the field are really waiting for the newly developed tool. Rescue workers are quite skeptical about importing new tools into their daily routine when the advantages are not explicit [KBR05a], [vHBR06].

An example of this phenomenon is given in a research paper by French researchers Dardelet and Darcy, in which they discuss an experiment which involved firemen carrying camera's on their outfit to provide medical experts outside of a crisis building with imagery of supposed casualties. The thought was that an instant checkup of casualties would make the rescue effort more efficient because then a far better estimate could be made of the planning of the number of needed ambulances, for instance.

Practice appeared to be a bit different. During crisis training, the setup was tested, and was found to be troublesome for the medical experts and the firemen. The medical experts were having trouble with the imagery and kept on asking for more details to the firemen, who were of course already doing enough other things at the same time. The firemen at the same time were quite bothered by the fact that they had to provide so much additional information to the medical experts. They had the feeling that the medical experts were taking over their work and during the execution of their job, there was a conflict of interest in what the medical experts on one side wanted, and on the other side what needed to be done according to the firemen. For example at one point during the training the firemen wanted to climb the stairs of a building on fire to see if someone had to be rescued there, but the medical experts first wanted the firemen to go and have a look for them at a person who was lying on the ground floor [DD03].

This example illustrates that before designing IT solutions for crisis response, much care and time must be put into the discussions with the users of the new solution and to make sure that it fits seamlessly into their regular duties. It's also very important that designers listen to the needs of the people in the field, instead of designing something out of the blue. Because especially in crisis situations decisions have to be made in split seconds, it is vital that rescue workers can trust on their instincts and not have to worry about significant changes in their regular routines caused by a new IT-gadget that they didn't ask for in the first place.

2.2 Related Work

In this section of the theory chapter we focus more on ideas found in literature and direct applications that are related to the design of MACSIM. For every key component in MACSIM we will consider first some general theory that gives some sense of direction how to categorize MACSIM with respect to concepts known in literature. Then we will give some practical examples of products and research projects that are related to the work and concepts in MACSIM.

2.2.1 Simulation

Gas Dispersion Simulation Models

In this section the most popular gas dispersion models will be discussed. As the goal of this thesis is not to give an exhaustive and complete descriptions of all available models in literature, and because of the fact that we did not use all the dispersion algorithms available during software development, we will confine to giving a short description of the types of gas dispersion models that are out there. For more details on these models, we refer to [CPR97], which is a standard work on all different types of physical modeling issues.

Before any particular gas model is chosen, we first need to look at the conditions of the air in which gases are released, because they have a large influence on the dispersion. The most important factors are: air temperature, wind speed, humidity, wind direction, cloud cover and time of day. Based on this, a category of air quality is chosen. There are different ways of qualifying the different types of atmospheric conditions. The most well known one and most used one is the Pasquill scheme [CPR97].

Based on wind speed (either in knots or in meters per second), cloud cover (in eights) and time of day and year, a Pasquill classification can be obtained, varying from A to F, with F being very stable and A being very unstable. When this classification letter is obtained, it is then possible to derive the essential parameters needed for dispersion formulae.

For the Netherlands, the Royal Dutch Meteorological Institute has made a general indication of determining the appropriate Pasquill stability class (Figure 2.1).

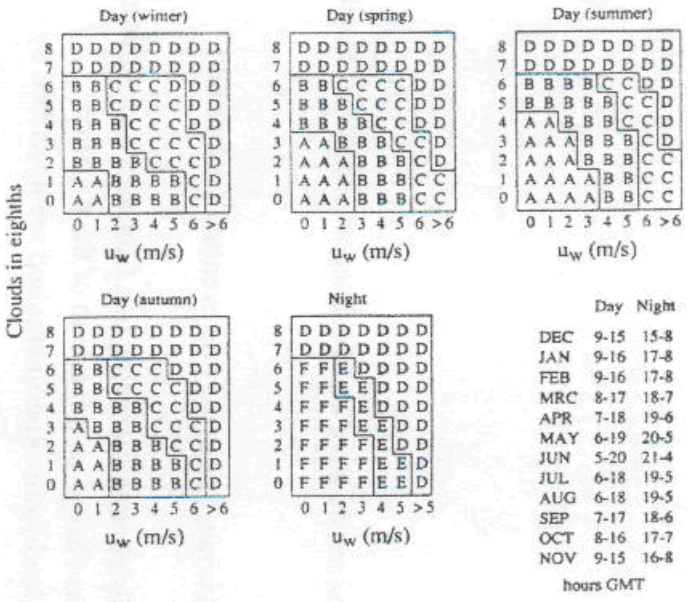


Figure 2.1: Pasquill Classes for the Netherlands [CPR92]

When we have determined this, we can continue choosing what kind of model we need to model our gas dispersion. There are 3 main categories of gas dispersion: Jet Streams or Plume Dispersion, Heavy Gas Dispersion, and Passive Dispersion. It depends on the type of gas that is escaping and where it is escaping from which type of model is the most appropriate one to choose.

Finally, most gas dispersion models use for the direction that the dispersion the terms "downwind" and "crosswind". When a gas is spreading, it spreads in two directions: the downwind direction, which is basically following the wind direction, and the crosswind direction, which is the direction perpendicular to the wind direction. In gas concentration calculations, the x parameter is always referring to the downwind distance, so the x-axis is the "downwind axis" and the y-axis is the "crosswind axis". This is important when using x and y coordinates in regular coordinates system; because then one first has to convert the regular coordinates to downwind/crosswind coordinates.

Jet Streams and Plumes

When the gas is being released with much force into the environment, it is best to choose the Jet Stream models. Most of the times the gas in this case comes out of a chimney or some sort of pipe. In the next figure the most important properties of jet stream dispersion are being described.

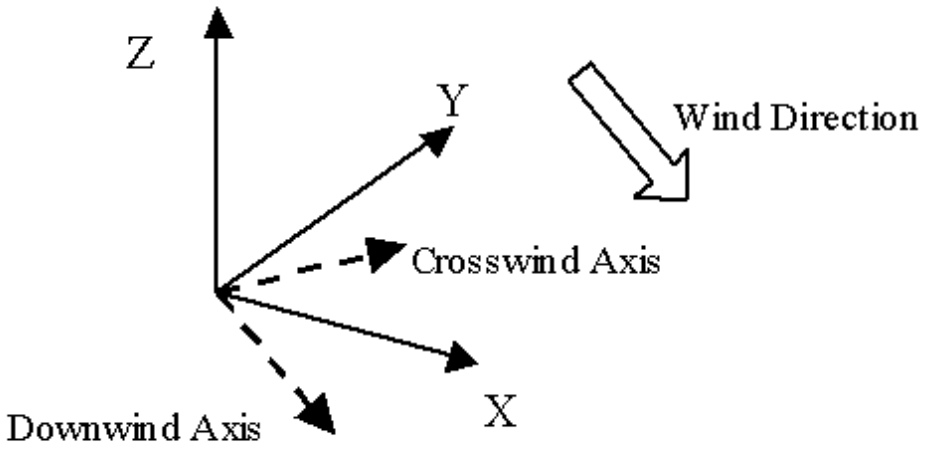


Figure 2.2: Downwind and Crosswind Axis

The development of this gas cloud can be described in three layers, as can be seen in Figure 2.3. The force with which the gas is shot into the environment dominates the dynamics of the core stream. This has its effect on the way the gas is spread into the environment. It is basically still a constant flow. In the shear layer, the main stream of the jet stream becomes less important and the small irregular air circulations become more important. In the fully developed region, the influence of the core stream is totally disappeared and the gas dispersion happens entirely through irregular air circulations or "eddies" as they are called in [CPR97]. This is called "passive dispersion" of which we come to speak later on in this section.

For describing a jet stream of gas, the Chen and Rodi model is most popular for concentration calculations. Because this model is not included in MACSIM, we will not discuss it here in detail. It should be said that this model could be added to a new design iteration of MACSIM, provided that the necessary parameters are given.

Heavy Gases

For the gases that are heavier than air, the initial behavior is somewhat different. The spreading of the gas cloud take place in 4 distinct phases: The Initial phase, the gravity spreading phase, an intermediate phase and then the passive dispersion phase [SAF], [CPR97].

In the initial phase, the gas behavior is dependant of the release conditions, such as source pressure, initial gas temperature etc. Then, because it's heavier than air, it slowly but certainly starts to "fall" to the surface (gravity spreading). It then still can be considered a coherent gas cloud. From then on, depending on the type of gas, the gas will eventually be dispersed passively to the surrounding air. Most of the behavior described here is highly dependent on the properties of the specific gas, because these different stages can follow each other in a very high pace but it can also take a long time before a state transition takes place.

Because of that, models for calculating the concentration of a heavy gas cloud are very complicated. The high number of parameters and dependencies are the main reasons for this. Nevertheless, in literature

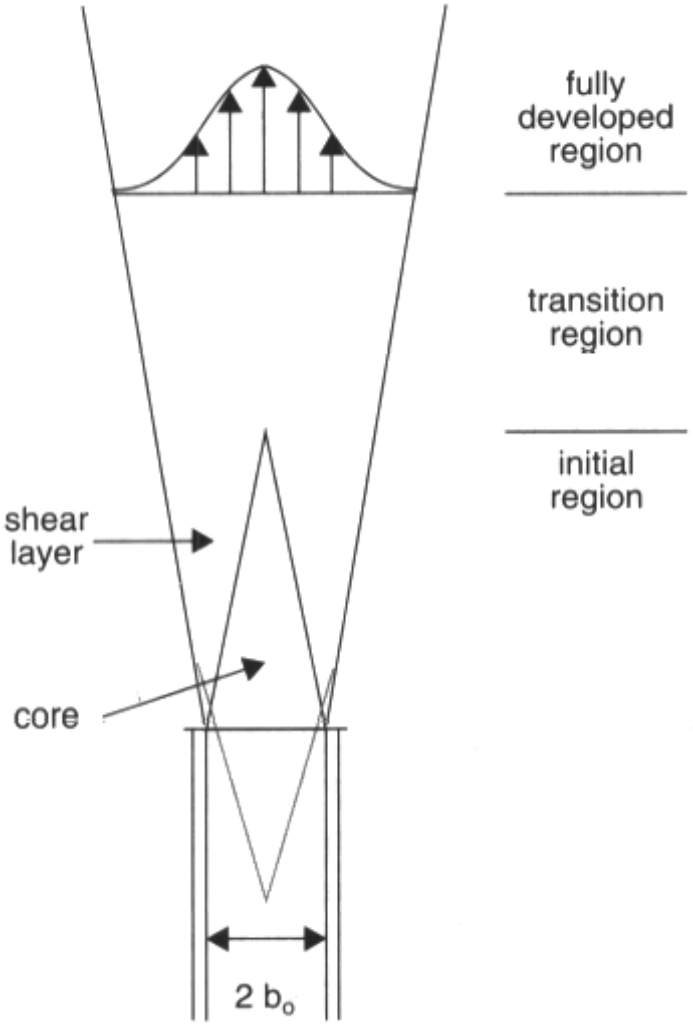


Figure 2.3: Plume Dispersion [CPR97]

there are some popular heavy gas dispersion models to be found, two well known ones being the models of Colenbrander (1980) and Kaiser and Walker (1978), which has been used in the DEGADIS model. The Colenbrander model is used for heavy gases in ALOHA, the Kaiser and Walker model was used for developing SAFER, both gas modeling software products will be discussed later on in this section.

Passive Dispersion

The third and most common category of dispersion is called passive dispersion. According to [CPR97], we can speak of passive dispersion, if the spreading of the gas cloud is being governed by atmospheric turbulence. This means that only external forces are responsible for the spreading of a gas cloud. Dispersion in this case is caused by small circular whirlwinds in the air called eddies.

In Figure 2.6 three different types of eddies are shown. Only in the case where eddies are significantly smaller than the gas (situation a), the gas cloud will spread uniformly through the atmosphere. In the other two cases the cloud will just move (b) or deform (c). In regular passive gas dispersion it is assumed that situation a takes place most of the time, because the size of the gas cloud is usually much larger than the size of eddies.

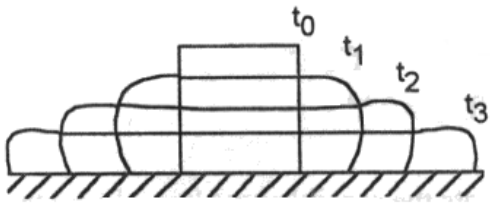


Figure 2.4: Heavy Gas Dispersion Stages [SAF]

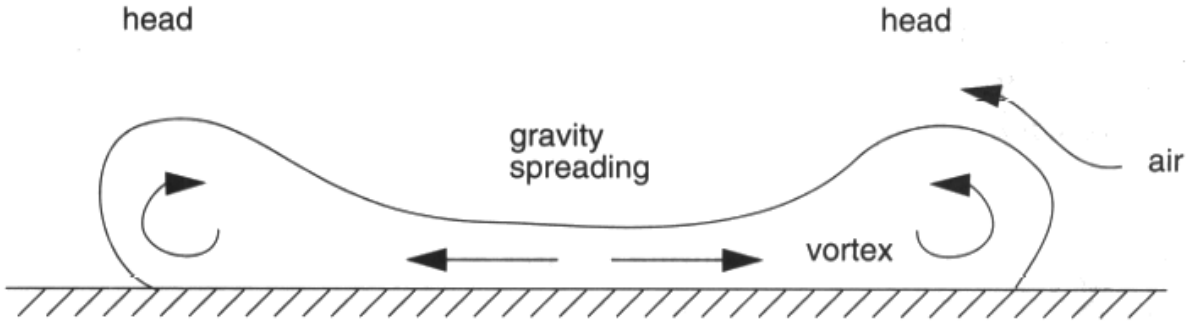


Figure 2.5: Heavy Gas Dispersion Final Stage [CPR97]

Because of the assumption of uniform distribution of the gas in 3D space, a Gaussian distribution formula for 3D is being used. To get a concentration on a downwind position x, a crosswind position y and at height z on a time t (time past since initial release time t_0). The following basic formula is being used for concentration calculations (for an instantaneous release of gas, not lasting too long):

$$c(x,y,z,t) = \frac{Q}{(2\pi)^{\frac{3}{2}}\sigma_x\sigma_y\sigma_z} \cdot \exp(-\frac{(x-u_a \cdot t)^2}{2(\sigma_x)^2}) \cdot \exp(-\frac{y^2}{2(\sigma_y)^2}) \cdot \exp(-\frac{(h-z)^2}{2(\sigma_x)^2})$$

In which Q is the total released mass. Sometimes Q is acquired by multiplying release rate q with relative time t. The parameter u_a is the wind speed and h is the initial release height. If the release takes too long it will become a continuous release, which looks like this [CPR97]:

$$c(x,y,z) = \frac{q}{(2\pi)^{\frac{3}{2}}\sigma_x\sigma_y\sigma_z} \cdot \exp(-\frac{y^2}{(2\sigma_y)^2}) \cdot \exp(-\frac{(h-z)^2}{(2\sigma_x)^2})$$

The power or weakness of the dispersion formula lies within the σ parameters. They are called the dispersion parameters, which can be defined in different ways. The sigma parameters are always functions dependent on x, y and z and some other parameters, either based on the air stability (Pasquill class), or on the time of year in which the release is taking place. The level of exactness is mostly determined by the way you wish to calculate σ_x, σ_y and σ_z. There are complicated ways to do this [CPR97], involving finite element calculations and time-consuming minimalization procedures. There are also some shortcuts to simplify the creation of the σ functions if you make some simplifying assumptions [CPR92]. The easiest way to get the above mentioned sigma functions is as follows:

$$\sigma_x = ex^f$$

$$\sigma_y = ax^b$$

$$\sigma_z = cx^d$$

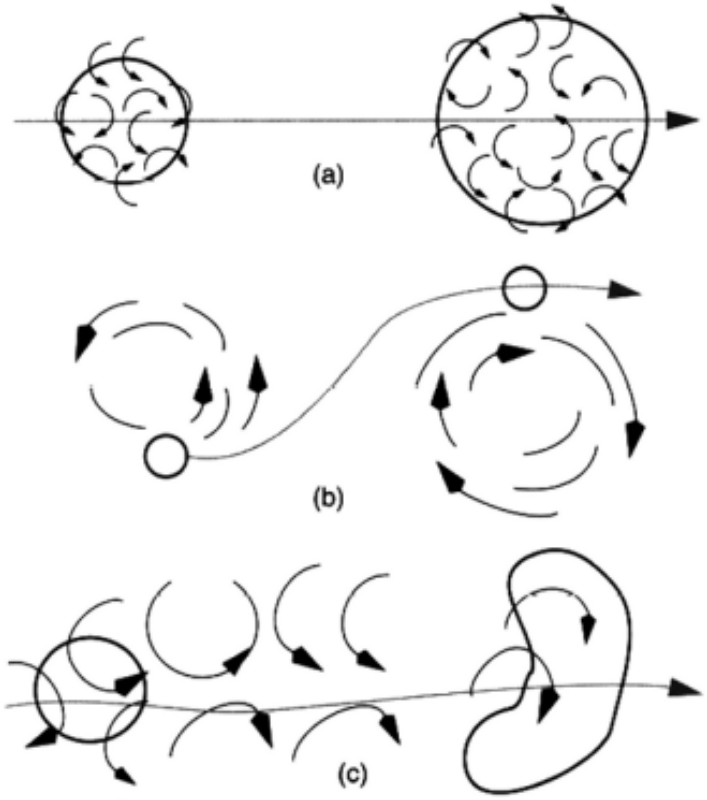


Figure 2.6: Turbulence Eddies [CPR97]

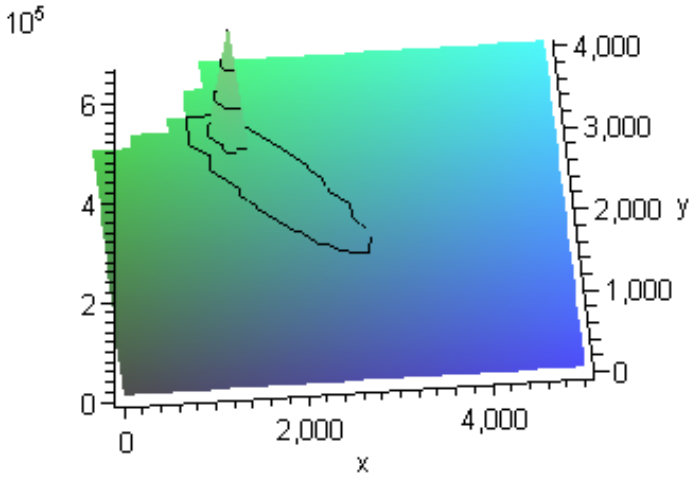


Figure 2.7: Dispersion Plot in Maple

in which the parameters a to f are based on the Pasquill stability class A to F. Exact details about this can be found in the Detailed Design chapter of this thesis.

The other complicating factor is the question whether the initial dimensions are negligible. If you choose to include source dimensions this will add correction factors to function (instantaneous release function dispersion) that require an integral calculus procedure of a numerical approximation function to solve. It remains to be seen whether the level of exactness that is gained by not ignoring the dispersion dimension is compensated by the addition of computational complexity, followed by the additional calculations.

Fire Simulation Models

In this section a general overview of fire simulation models will be discussed. To be precise, a general overview of that type of models that are available to describe the development of heat radiation and the transfer of heat in case of a fire. This immediately shows the limitations of this section. The models known in literature are very wide in variety, and in circumstances, so to give a complete summary of all models and the specific cases in which they can be applied could become a literature survey on its own. Therefore we will suffice by giving a general classification of what different types of fire models there are available, should somebody want to include a specific model into a new version of MACSIM. For a complete description of models and the corresponding formulae, please see [CPR97] which has a comprehensive overview of all available models out there.

The fire models can be divided into three main categories: Semi-empirical models, Field models and Integral models [CPR97]. Semi-empirical models are very fast in use and usually are meant to describe the general shape of a fire and the flow of heat throughout the duration of a fire. Some of those models assume that the fire starts in a point, and from then on starts to spread, other models assume that an object is burning and that the heat is coming from within that object. They are most commonly used in crisis assessment procedures because of their calculation speed and because they are easy to understand. The downside of these models is that the parameters used in these models are based on acquired data, on a specific type of fire under specific circumstances. This means that one should be careful not to use this type of model for a type of fire that it's not designed for.

Field models are very complicated mathematical representations of fires. These models are very complex and have very long calculation times. They are usually only run on large supercomputers with a large computational capacity. This makes them not very useful for inclusion in a real-time crisis simulation, because the computation of all the effects will just take too much time. The good side of these models is that they are very generally applicable.

In the grey area between Semi-empirical models and field models there are the Integral models. They try to get the best of both worlds. They try to reduce the computational complexity of the model by using integral functions, while calculating the same kind of quantities as can be done with field models. The reduction of computational complexity makes them faster, though they still leave a lot of computation time, which still make them difficult to use in real-time software simulations. It still needs to be said that these models are more general and therefore can be applied to more different types of fire than is possible with semi-empirical models.

The general conclusion is that if a realistic fire model should be included inside a real-time simulation, the most preferable type is a semi-empirical one, mainly because of the reduced computational complexity.

Software Examples

In this section we are going to look at software examples in which several features that are required in MACSIM are implemented and can be used as good reference points for the design of MACSIM.

ALOHA and MARPLOT

The US government organizations EPA (Environmental Protection Agency) and NOAA (National Oceanic and Atmospheric Administration) have made a very easy-to-use gas dispersion modeling tool called ALOHA (Arial Locations Of Hazardous Atmospheres). It is able to model chemical releases and has some advanced features in modeling [NOA04]:

- Evaporation of liquids that have leaked on the ground
- Flow of chemical clouds until an hour after initial release
- Moisture condensation within a gas cloud
- Gas or liquid emissions from tanks or containers under pressure
- Difference in modeling between gases heavier or lighter than air.

ALOHA has been designed for people with crisis response duties, so after an incident they can get a fast overview of what is going on and what will be the situation in the upcoming hour. For this timeframe it can predict release rates and gas concentration on any given (x, y, z) coordinate relative to the origin or in GPS format [EN04]. It is able to model the dispersion behavior of over 1000 different chemicals.

The only thing that the user needs to do is to give up some basic parameters about the situation and the location concerned. The following set of pictures describes the setup of a simulation in the ALOHA software. After the setup of the simulation, the system can create a footprint for the area surrounding the incident location. A footprint is a curve that is plotted around the incident location that is based on the concentration levels that can be hazardous to the general public. It is basically a set of isocontour curves in which each curve represents a different concentration level (See Figure 2.8).

It is also possible to get the concentration curve for the next hour from a given (x, y, z) coordinate shown in a diagram. This (x, y, z) can either be given as a relative coordinate or as fixed coordinates. Also the dangerous concentration levels for that the particular gas are shown in the plot (Figure 2.9). In combination with another program made by NOAA/EPA, MARPLOT, ALOHA is capable of plotting the footprint on top of a map of the area concerned (Figure 2.10).

Unfortunately, MARPLOT is a piece of software that uses maps that were designed by the US Census bureau [ENotC], and so there are only US-based maps included in MARPLOT and only US based scenarios can be shown in this way. Besides that, ALOHA has some other disadvantages for being a good simulator. The most prominent one being the fact that it doesn't give real time information. It is a very fast calculation tool to determine an area involved in a gas escape, but it does not calculate gas concentration real-time for every location in the area. It therefore cannot be used as a piece of software that can generate events in real time. Another downside of ALOHA is that it does account for geographical properties of the area concerned [EN04]. It assumes every landscape to be flat. In the Dutch situation this will probably not be a great problem, but as there are only US maps available in inside the ALOHA/MARPLOT combination, this is not a minor detail that can be easily overlooked.

Safer Realtime

A program that is especially dedicated to real time simulation is the program Realtime, which is made by Safer Systems. Once a branch of DuPont de Nemours, one of the leading chemical companies in

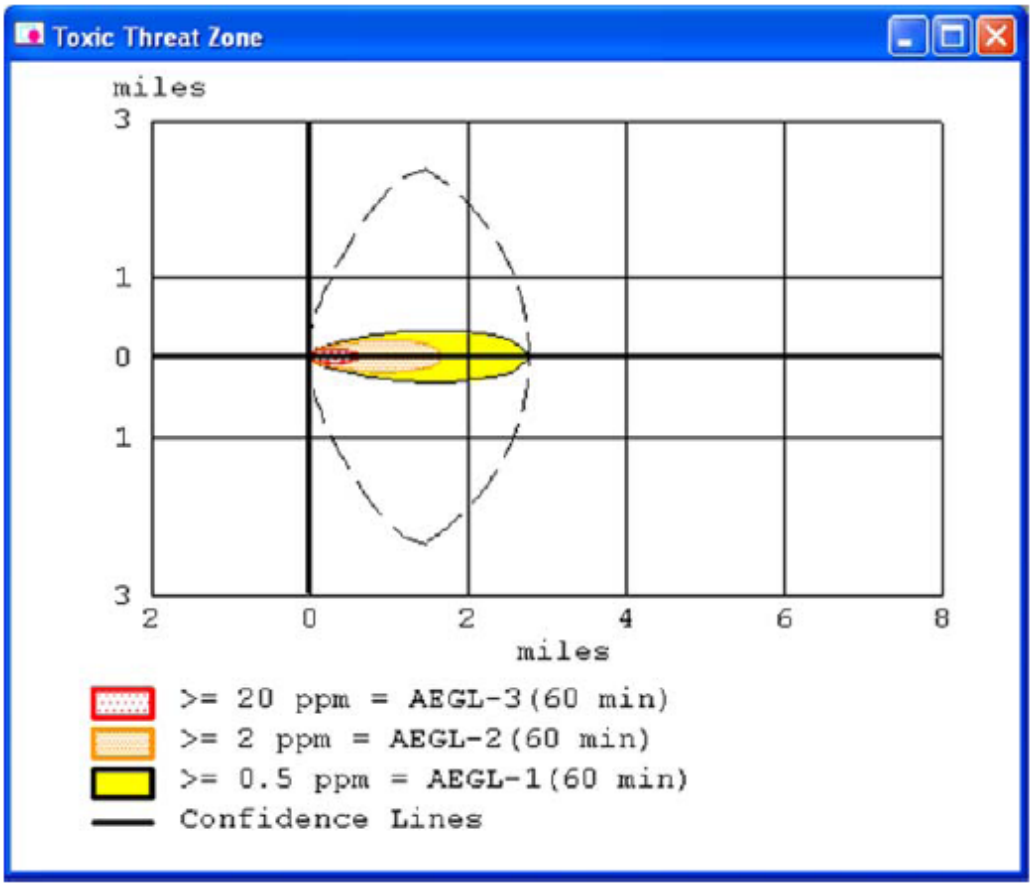


Figure 2.8: ALOHA Footprint [EN04]

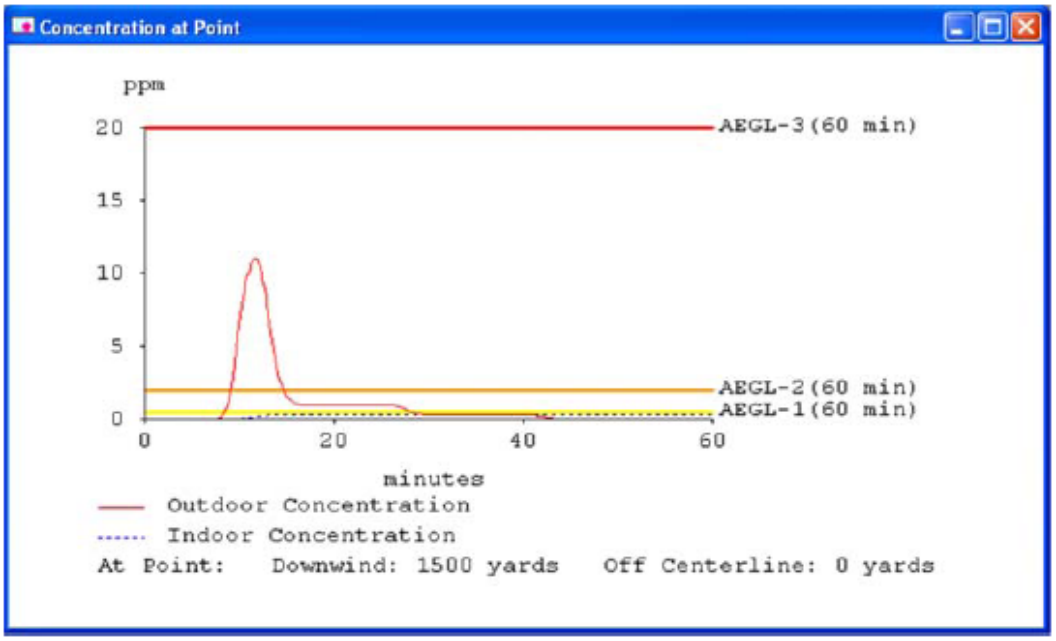


Figure 2.9: ALOHA Concentration Curve [EN04]

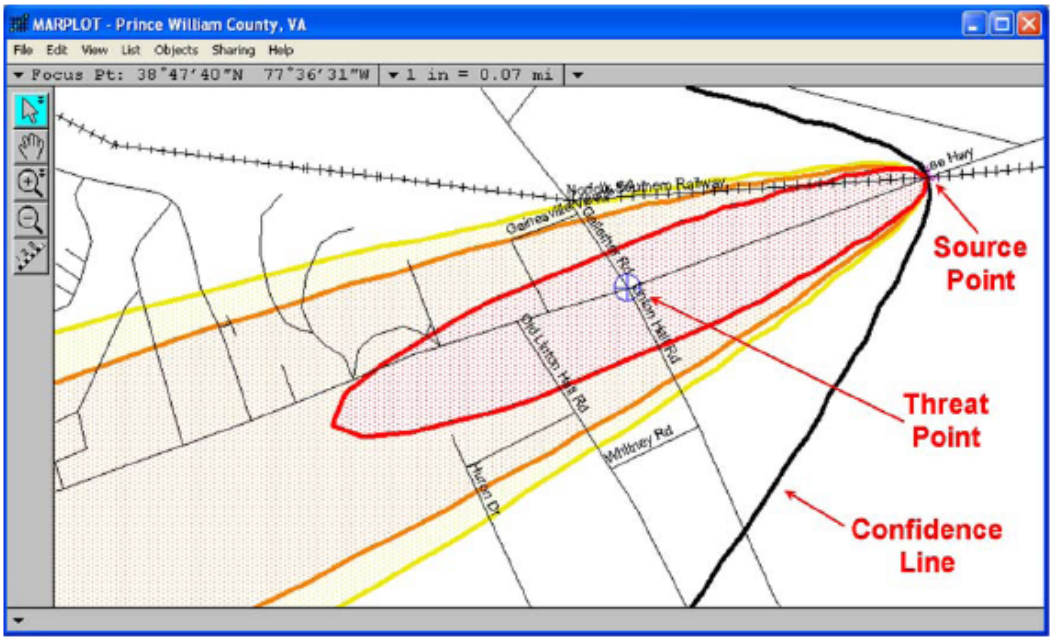


Figure 2.10: ALOHA and MARPOT Interaction [EN04]

the world, it is now an independent company that is dedicated to chemical risk estimation and chemical simulation. Realtime is a program that is used by a lot of chemical companies for determining the consequences of an incident that just happened. It can also be used as a simulator [SAF].

Because of the fact that it's a commercial product, it is usually tailor-made for the customer. This means for instance that the area in which the simulations take place, is always the same, namely the area in which the factory is located. Therefore more location-specific information can be added to the models. The program gives direct access to meteorological data that is coming from real sensors, but it's also possible to give the meteorological data as manual input. It is capable of producing footprints on the map that is updated in real-time as the simulation progresses. The circles on the picture indicate where the center of the gas cloud will be, according to the current wind speed (see Figure 2.11).

The models that are used are a Gaussian-based model for gases lighter than air, and Colenbrander's

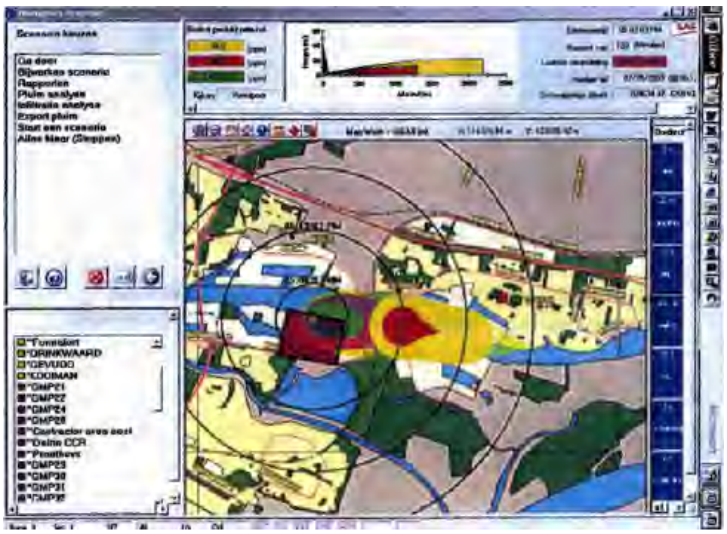


Figure 2.11: Safer Realtime User Interface

dense gas dispersion model for heavy gases [SAF]).

Crisis Simulation using Waypoints and Crisis Scenarios

In the first half of 2005, two French exchange students, Florian Haradji and Hervé Ducoutrieux worked at the MMI lab at TU Delft on a project to create a crisis simulator based on a car traffic network [DH05]. The concepts of the car traffic network were based on the findings of Bogdan Tatomir [TR04]. The software they made contained a lot of interesting concepts that were a substantial reference during the design of MACSIM.

The simulation environment contains waypoints for storing the data. Waypoints are points placed on a location (x,y) that can contain any type of information concerning that particular location. In this particular system it contains information about roads, because each waypoint in this case is placed at the beginning of a piece of road in the car traffic network. The effects of a simulated crisis are propagated through the waypoints, as can be seen in 2.12.

The updating process in the application is supposed to be memory efficient, so for updating waypoints it should be the case that no unnecessary updates should take place, and no irrelevant waypoint properties should be stored in the waypoints. The intelligent updating takes place in such a way that during each update instance the software component responsible for updating the waypoint get a list of waypoints that need to be updated. This saves considerable time and resources if there are a lot of waypoints properties to be updated.

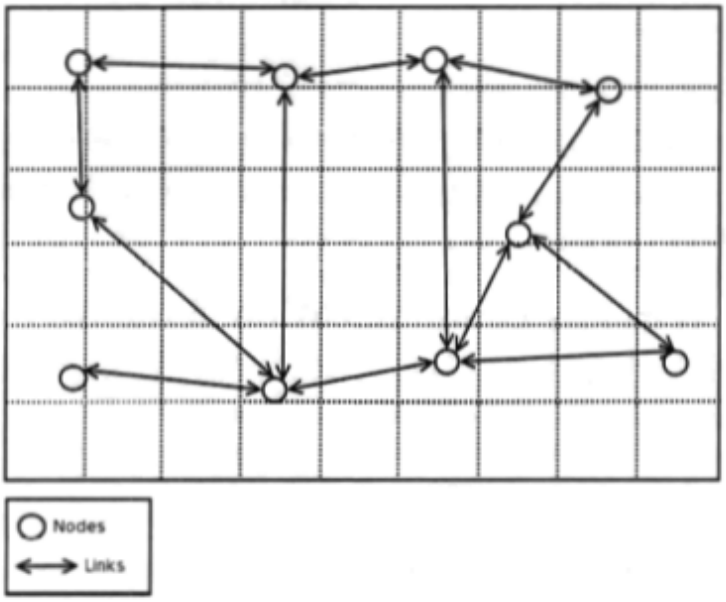


Figure 2.12: Waypoint Design of Haradji and Ducoutrieux [DH05]

From a designers point of view the program had the useful property that the update algorithms were programmed generically. This meant that if later on it is needed to add new properties in the waypoints it should be easy to do so. Also if it is necessary to replace existing algorithms it should be done in such a way that it can be done so that the rest of the program is not affected by it. Because the scenarios used in this program are script-based, it must be possible to make scripts. Programming files in the Python scripting language generates scripts. It also means that the events that are unfolding are time-based, to that a scheduler scheme is present to make sure every event is being started at the right time.

All the properties described above are considered to be essential properties for a system like MACSIM. In the Design chapter we will therefore often refer to this application as far as it comes to building a world model and updating scenarios.

Applied Simulation for Training Purposes

At the MMI group, Dr. L.J.M Rothkrantz is currently working on a research proposal that focuses on the development of tools that can help crisis management and response agencies improve their cooperative skills and improve the possibilities for training. Also the proposal aims at building tools that can simulate disasters. The proposed project should be conducted at the DECIS lab and is called (ISDM or Interactive Simulation in Disaster Management) [Rot06].

The idea is to bring a lot of expertise on crisis simulation and training together in one community within the DECIS environment. Tools will be implemented and designed on a multidisciplinary basis. The goals of the project are:

1. To develop new communication and decision structures of intelligent sensors, networks and knowledge systems, integrated in a coherent intelligent system.
2. To develop a crisis simulation system with a high level of realism, with a link to real disaster objects and communication structures. Through this system new methods of coping with a crisis and new disaster management plans can be designed and tested. Eventually the system can also be used for training crisis situations with real human participants.

The system defined in goal 2 should be designed in such a way that real life objects and their digital simulated counterparts can be exchanged if necessary. If for instance real telephone communication can be provided but not real meteorological data, the system should be adapted so that only the parts that are not available in real life are provided in digital form by the system. The core elements of a system proposed by Rothkrantz are a disaster generating device and a virtual world. These components are also available in MACSIM.

When crises are simulated, for instance for the exams to become a chemical expert (ROGS), the simulation of a crisis is divided into four phases [vHBR06], [NBB02]:

1. Investigation, complaints are coming in and it needs to be determined what is going on at what location and if experts should go to look what is going on that location.
2. Investigation at the spot. Experts at the spot observe what is going on and based on that a response plan is being determined: this could mean things like evacuating an area or calling extra assistance.
3. The crisis response plan is being executed. The objectives are that the danger for the community should be gone and there are no more people in danger.
4. Cleaning up afterwards. The damage caused by the crisis is estimated and a plan should be made to clean everything up.

The best example of a crisis simulator designed for training found in literature is CMSMAP, a crisis management system, built for the Maritime and Port Authority in Singapore [AHGG02]. It represents a singular integration of ship's bridge simulator hardware and software, numerical models, and emergency response software. It is able to simulate different types of crises that can be expected inside a large port

environment, such as oil or chemicals that are lost, nuclear threats, and search and rescue operations. It is then possible to give management responses and do resource investigation in real-time as a reaction to the information that is coming in.

2.2.2 Multi-Agent Systems

Multi-Agent Systems are computer systems that are composed of multiple "autonomous agents" that together accomplish something that would be more difficult if it had been one basic program or just one single agent. Most of the time they are software agents, i.e. independent pieces of running software that accomplish a certain task. Though literature has a lot of definitions available for defining what kind of agents this should be, we believe that an exhaustive survey of what is and what is not a software agent is not within the scope of this thesis and therefore will not be discussed. Very interesting theoretical surveys of different types of agents and their corresponding systems can be found in [RN95] and [Woo02].

The most important thing to know about the definition of an agent for MACSIM is the fact that an agent in our system view is an autonomous (independent) piece of software that observes the environment, interprets the input and reacts to it by executing certain actions. This comes close to the definition of a reactive agent used in [RN95].

An interesting question for this section is whether the solution that we present in MACSIM given our definition of an agent, can be considered a Multi-Agent System. We want to build a system in which agents are walking around in the world, sensing things and communicate about it to other agents or the crisis center. This could make it a Multi-Agent System, except for the fact that the crisis center is the only agent that makes command decisions. The entire group of agents in the field does the reporting and gathering of information, but the decisions to initiate a certain action are made by the crisis center. This is an accurate model of the hierarchical way in which information in real life crisis situations comes in at the crisis center (or COPI). Decisions are made there, and although the people in the field are locally responsible, the final command is still in the hands of the crisis center.

The agents in MACSIM are certainly autonomous, because they decide on their own whether or not they should report something to the crisis center. It is therefore undetermined whether they report anything at all and if they do report something, when they are going to do so. This makes them more than just "sensory input generators" for the crisis center. The crisis center cannot decide anything without the information support that it receives from the agents in the field. There exists a clear dependency between the two agent categories.

In most multi-agent applications, the agents independently go to research something and start to communicate with each other and make decisions on their own to achieve some sort of goal while dividing the task that is to be done into small sub-tasks that each agent has to perform individually [Woo02]. In the case of MACSIM, the agents walk around in the world, and do sensory tasks for the crisis center, which does the actual crisis assessment. It is in therefore an example of a "coherent intelligent system" proposed in [Rot06]. They collaborate to acquire the information, but they do not collaborate in making decisions for the collective of agents. Therefore MACSIM can only to a certain extent be considered a collaborative Multi-Agent System, but it is a coherent intelligent Multi-Agent System consisting of multiple information gathering agents and one decision agent. Thus it has some aspects of a single-agent architecture, but also of a multi-agent architecture.

Agent Middleware

Whatever kind of agent system is used, the agents have to be able to communicate. This is done via Agent Middleware. This is software that is dedicated to administering the agents in the system, and facilitating transport of messages. A very popular Agent Middleware is JADE (Java Agent Development Environment) that has been developed by Telecom Italia Laboratories [BPR01] and is compatible with

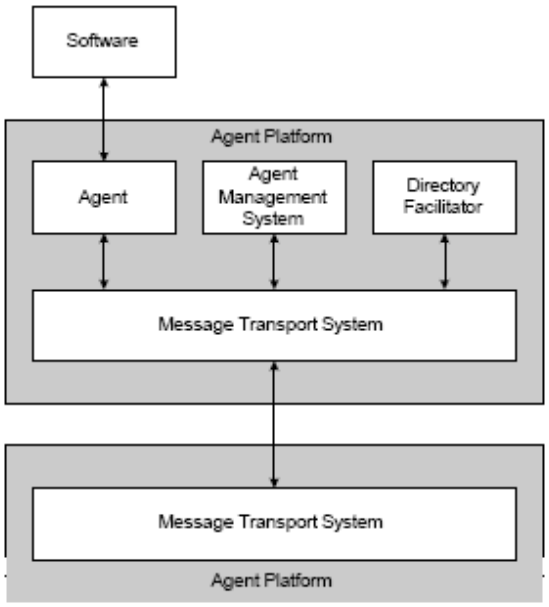


Figure 2.13: FIPA Platform Architecture [FIP02]

FIPA (Foundation for Intelligent Physical Agents) standards for Multi-Agent Systems [FIP02]. FIPA is a standardization organization for multi-agent systems which provides standards for agent management and agents communication protocols (more on this in the section on protocols).

Compliance to the FIPA standards for Multi-Agent Systems means that it has an Agent management system (AMS). This is an agent that controls which other agents are on the platform at every point in time and what their status is. Each new agent should register with the AMS. A Directory Facilitator is an agent that provides a lookup function for services-related requests (Yellow Pages functionality). An agent can subscribe to this service if it wants the other agent to know that is delivers a certain service. So any agent that calls the DF can ask which agent provides a service A and then the DF responds by giving the ID's of the agents that do so. Also it has a message transport system that controls all message passing within the system. Messages are sent via asynchronous message passing.

Because of the extensive use of Java-based software at the MMI lab, and the popularity of JADE within the agent community, it was decided that MACSIM should run on the JADE platform. The main reasons why JADE is so popular is because of the fact that JADE Software is still very actively updated (version 3.4 dates from march 2006) and it is open source software, and therefore free for academic purposes. Also there are a lot of documentation and support tools available for JADE, such as the JADE ontology plug-in for Protégé ontology software and online tutorials for creating JADE-based programs. However, there are other examples of agent middleware, like Tryllian ADK (closed source), FIPA-OS (not as widespread as JADE) and Cougaar (makes use of a distributed blackboard for inter-agent communication). For other examples and a comprehensive overview of more agent middleware and their specific properties, please refer to [Gro04].

Finally we will give an overview of some related Multi-Agent Systems based on JADE. In the MMI Lab at TU Delft one of the examples of Multi-Agent Systems that were based on JADE was PITA by Martijn Beelen [Bee04]. An example found in literature of a JADE-based crisis simulator was made by the team of Ben-Saoud [SPD+05] at RIADI-GDL Laboratory ENSI, in Tunisia.

PITA

Designed as a travel planner for mobile devices, PITA (or Personal Intelligent Travel Assistant) is a system that enables the user through updates of delays in departure times of trains to calculate the best train connection for a traveler. The delay updates are given to traffic control servers through wireless communication, after which they are synchronized and users of PITA are being informed about any changes in their proposed ideal schedule. The updates of train delay information were processed by a distributed multi-agent platform, based on JADE.

Agent-based Crisis Heuristics

The goal of Ben-Saoud and his team was to model the problem of large-scale accident rescue effort by applying an agent-based approach [SPD+05]. The team was largely working on crisis situations with a large number of victims. The focus point is to ensure rescuers can save the largest number of victims as fast as possible by optimizing both their human and material resources. The foundations of the research were real life observations and rescue plans. Ben-Saoud and his team succeeded in building a generic and interactive user-friendly simulator for crisis simulations. This simulator was programmed in a JADE environment with added visual monitoring components. Agent-based modeling and simulation within a virtual environment enabled the team to develop and test a large number of different crisis scenarios for different rescue organizations. The goal was to eventually design new rescuing strategies based on the testing.

2.2.3 Agent and Human Communication

In MACSIM, the agents have to communicate with each other to keep each other informed about crisis situations. Also the communication has to be represented in such a way that the meaning of the sent messages is clear. There are different ways of achieving this communication between software agents. In this section first some general practical examples from literature are given, then some examples developed at TU Delft are presented, and then the method of communication that we used in MACSIM is being explained, including the ontology and the visual interpretation of that ontology.

GeoMIP

GeoMIP (Geographical Multimodal Interface Platform) is a demonstration program that has been developed by Penn State University to achieve an easy user interface for crisis management decision makers that is based on speech, and gestures [ARI+04]. Therefore the decision makers can retrieve the information they want for in this case a GIS system, through gestures and speech, that is processed by the system and is then converted via Multimodal Fusion to a communication protocol with a GIS server that produces the right set of GIS map data. The gesture-based communication is achieved trough a rule-based dialog schema that is being used to get the best gesture fit. The features in this application could be an excellent addition to the simulation-based training proposed in the ISDM project and in new design iterations of MACSIM.

NIEM

The US Government in 2006 has developed an content exchange model that is called NIEM (National information Exchange Model). It can be used for the sharing of information, especially the information that is already exchanged in existing information systems dealing with safety-related subjects [DD06]. NIEM attempts to create a set of concepts that are semantically unambiguous and commonly accepted so all the agencies involved in using the NIEM standard for processing data have a standardized corpus of concepts to process safety-related content. This is achieved by creating standardized XML schemas that can be applied for exchanging information about certain crisis and safety related topics like immigration,

environmental dangers and infrastructure protection. Because of the implementation in XML schema documents, these protocols can be easily integrated in existing software project that have to exchange information about these subjects.

CHIM

In the DECIS environment the MMI lab is involved in the CHIM (or Computer Human Interaction Modeling) project. In environments such as a crisis center, it is important that the right information is available at the right time [DEC06]. Therefore in the CHIM project the focus lies on Multi-modal input interfaces for crisis tools, for instance speech processing, processing of hand-written input. Nonverbal communication is also taken into account. The output can be spoken or visual. These input devices and means of output should of course be connected to crisis management and response software. The main focus of the CHIM research is to find out whether fundamental mechanisms of human information processing can be revealed that are relevant in situations with a lot of stress and in situations where critical decisions are made. Dialog modeling is one of the most important parts of the research.

Distributed Blackboard Communication

An often-used method to communicate between agents is the method of a distributed blackboard. In this scheme messages are stored at a shared piece of memory that multiple agents in a certain environment can see [Kla05]. There are two different types of distributed blackboard systems: distributed blackboards with a centralized setup and blackboards with a distributed blackboard setup. The centralized variant has an advantage that there is not too much communicational and spatial overhead because of the fact that the messages are only stored at one place. The bad side of this is that in case the central storage server breaks down, everything is lost. This was exactly what would happen in a 9/11-like scenario. This immediately explains the interest in a decentralized variant of a distributed blackboard system. In this case the blackboard system is broken down into pieces that are stored on different storage locations. This means that fallout of one location does not mean all information is lost. It means, however that for keeping all parts of the blackboard up to date, extra messaging traffic between the different parts of the distributed blackboard is necessary. This structure is very useful for Wireless mobile ad-hoc networks, or MANETs.

At MMI lab, there were two recent projects involving MANET communication, TICS, by Paul Klapwijk [Kla05] and ManetLoc by Marcel van Velden [vV05]. Both applications focus on the sharing of knowledge of a partly unknown world in which new upcoming details of a certain developing situation are shared through sending and propagating messages though a MANET.

TICS focused on facilitating the sending of messages as part of the ISME program made by Paul Schooneman [Sch05], in which icons could be placed on a map via a PDA. This updated set of information on the map should then be updated on all the PDAs in the network. ManetLoc shares the topological information of an unknown building in which the total known structure of the building is being updated each time a new piece becomes known. The updated map is then "shared" and merged into a new map.

FIPA-Based Communication

One of the most used methods of communicating between agents is the communicating via the communication standards that FIPA provides. Besides standards for a Multi-Agent System, FIPA also devised a set of communication protocols to standardize different types of conversations that agents might have, like for instance a Query, an Inform or a Subscription for information [FIP02]. These different types of communicative acts are also called performatives. FIPA also has devised a standard structure for sending messages trough a Multi-Agent System, called ACL (Agent Communication Language). This

means that an ACL-compliant message has a certain structure with slots that contain information about the message. The most important slots are the slots for a sender, one or more receivers, the FIPA performative to be used, a reply-to slot a reply-with slot and of course the slot for the content itself. this is shown in Table 2.1.

Table 2.1: ACL Message Slot Examples

Parameter	**Features**
performative	What kind of message is it
sender	Who is sending it
receiver	Which agent will receive it
content	What is to be sent
reply-to	Subject of a message to which a receiver can reply
reply-with	If the message is a reply, the message name to which it is a reply is mentioned here

For MACSIM, the contents of ACL-compliant messages will be mostly information about a developing crisis that the agents in the field report to other agents. We therefore needed a basic set of concepts that contained all the different kinds of subjects that a crisis management agent in the field might want to discuss. This corpus of concepts was eventually found in a world model designed by Siska Fitrianie [FR06], which contained a lot of concepts related to crisis management and response. The great advantage of Fitrianie's corpus of concepts (or ontology) was that it was designed inside the knowledge engineering software Protégé, which allowed fast conversion to Java objects, and even better, a FIPA compliant ontology. Therefore we had a very wide range of concepts at our disposal for the agents to discuss.

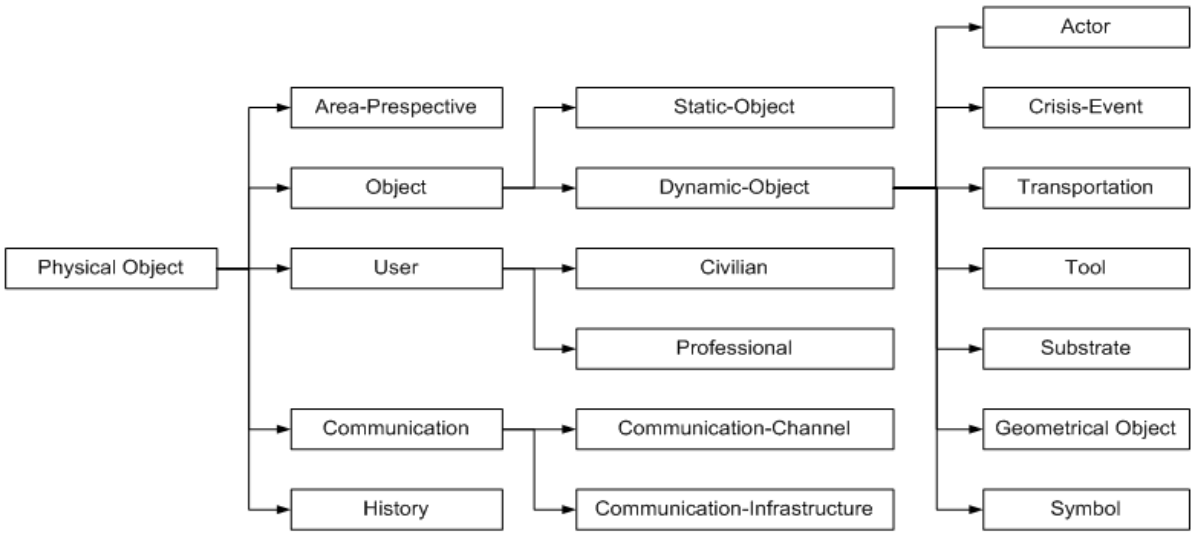


Figure 2.14: Part of the Crisis Ontology Taxonomy [FR06]

Unfortunately the only concept that was missing in the corpus as such was a means to give a specific order to agents. This was later added in the implementation process as a Directive (More about this in

the design chapter).

Finally, based on the messages that could be created by Fitrianie's ontology, another useful application was devised at the MMI lab. Iulia Tatomir designed an application called 3MNews (Message-based MultiModal News) [Tat06], which generated multimodal output for a news report, being an informational location-specific SMS and a map based on one single message created by Fitrianie's ontology. The map contains icons representing the reports that come in from a certain location in the world. This visualization was such an interesting feature that it was decided to integrate that visual component into MACSIM for the representation of an updateable map inside a crisis center.

2.2.4 Agent-based Reasoning and Decision Making

First we will discuss some crisis response topics related to AI and reasoning found in literature. Then we will discuss theoretical aspects concerning the nature of the agents in the MACSIM system and decision making inside the reasoning parts of MACSIM.

Yuan and Detlor describe in [YD05] what the main tasks of an intelligent mobile Crisis Response system should be. They distinguish the following tasks (not necessarily sequential):

1. Monitoring and reporting - through notices of the general public and intelligent sensor networks participants should be informed about crisis at hand.
2. Identification - Quickly getting a clear picture of what is going on.
3. Notification - Determining what crisis response agencies should be contacted and how.
4. Organization - Describing missions for the appropriate crisis response agencies.
5. Operation - coordinate tasks between different crews.
6. Assessment and Investigation - checking causes of the crisis.

Most of these tasks will be part of the first version of MACSIM. Initially, not all of these functions are implemented in the first prototype version of MACSIM, but the rationale for all functionalities given in [YD05] is a good validation for the above functionalities, therefore it is recommended to implement all of the above actions in later versions of MACSIM.

Another set of guidelines for designing intelligence inside MACSIM can be found in an article by Bui and Lee [BL99], that describes guidelines for designing a specific framework for agent-based decision support systems. It consists of determining what the problem is that the agents need to solve, specifying its functionality, determining the agent's characteristics and behaviors and implement the agents inside the environment in which they should function. Most of this will be discussed in the Design chapter of this thesis.

An example of using AI inside crisis systems or for instance intelligent sensor networks as proposed in [YD05] or [Rot06] is a fire detection system using diagnostics based on infra-red sensors [AFR]. This device, created by Advanced Fuel Research Inc. is able to determine whether gas particles passing through infra-red sensors which determine the concentration of a series of organic gases in the environment are part of a fire or not. It does this based on a trained neural network that gives the concentration inputs of all different gases recognizable and as an output the probability of the situation being a flaming fire, a smoldering fire or another environmental source. Not only smoke detection based on infra red sensors, but also based on visual processing is possible. Based on visual signal processing, detection systems like the one described in [Ebe99], that is designed by Siemens Research and Development can distinguish fires and smoke based on image-filtering processes and fuzzy logic.

MACSIM Agent Characteristics

In MACSIM the agents move around in the world and inform each other about what is going on. This behavior is based on some assumptions about the behavior agents. In literature there exist criteria to determine to which extent an agent possesses intelligence and what behavior belongs to a certain level of intelligence. One of the definitions of agents is that of a simple reflex agent (the simplest form of agent design), given by [RN95]. A reflex agent observes the environment through sensors, and then based on condition or action rules is able to decide what should be done (see Figure 2.15). For the agents in the world that observe things, this is the most appropriate design scheme. The only thing they need to know is what is going on at their location and what kind of messages they get from other agents. Based on that they decide what should be done.

For the crisis center this is a little bit different. It decides what kind of decision to make based on an

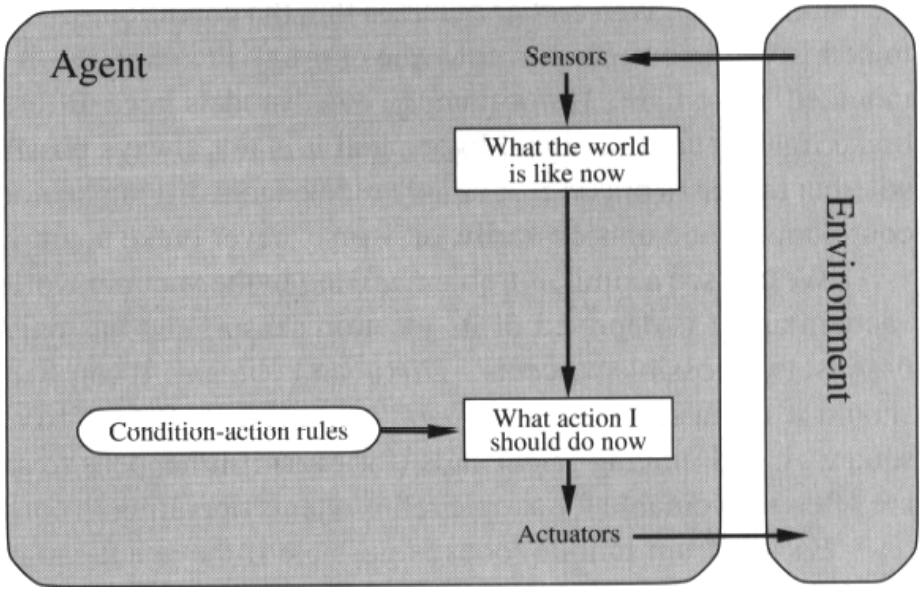


Figure 2.15: Simple Reflex Agent [RN95]

interpretation of the crisis that is going on in the world. Based on all the information received in the past and the knowledge that is available (some of this based on expert knowledge) the decisions are made. This means that the crisis center has some sort of model of the actual world available for the foundation of the making of the decision. Of course it also makes use of condition/action rules to execute actions. From the definitions mentioned in [RN95] the definition of a model-based reflex agent comes close to what we were looking for in the design of the agents in MACSIM (Figure 2.16).

An advantage of the structure of a reflex agent (whether or not reflex-based) is that it can be described in such a way that it is easy to represent it as a JADE agent. For more about this see the Design chapter.

Jess as Rule Base

To make the decisions inside agents during the decision process, there have to be rules for the agent to use to base its decisions on. These rules have to be interpreted and checked. Therefore a rule base was needed. Rules in a rule base will be matched to the facts that are inside the rule base. When a certain rule applies, it is said to fire. It basically means that whatever needs to be done should the precondition (left-hand-side or LHS) of the rule occur, will be done (i.e. the right-hand side or RHS will be executed). Rule bases have a great advantage over simple sets of if then statements in program code, in the

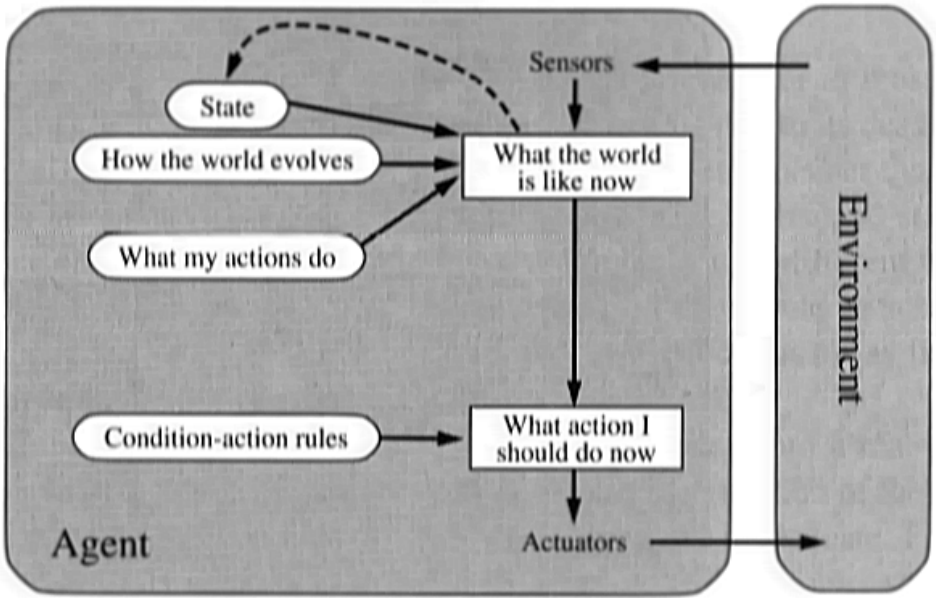


Figure 2.16: Model-based Reflex Agent [RN95]

sense that they are able to process rules in a non-deterministic fashion [FH03]. This means that it is not certain in which order rules will be activated. This makes the code much more efficient than a whole set of if-then constructs inside programmed code. The only thing that needs to be done is to run the knowledge base on the basis of the current input. Then the corresponding rules will fire and based on those rules that are fired the agent will perform certain agent actions.

Because MACSIM will be Java based, Jess (or Java Expert System Shell) was a good choice. Jess was developed at Sandia Laboratories and is free for academic use. For the full version of Jess a license is needed that can be freely acquired from Sandia. Jess is Java-based and can therefore be used very easily in combination with Java code. The language in which rules must be programmed is close to the expert system language CLIPS, only with a few modifications [FH03]. Jess is a rule base that is programmed in Java and can run as a standalone program, but the full Jess version also contains a Jar file so the Jess functionality can also be called from outside the Jess environment. Another advantage of Jess is that besides regular facts, it can also treat Java objects as if it were facts. So inside Jess it is possible to make rules that are based on certain properties of Java objects, as long as they have the functionality of the Java Beans 1.1 standard (http://java.sun.com/products/javabeans). For Jess this basically means that each object must have get and set methods for the properties to be recognized inside Jess and when objects should be manipulated inside the Jess rule base, a PropertyChangeListener should be added. The reason that this property of Jess is convenient is because it allows us to use the Java-based ontology by Siska Fitrianie to be used as facts inside the Jess rule engines.

With the agents having a rule base, they should of course also have rules at their disposal. The rules sould be based on real-world knowledge to make the decision process of an agent inside the simulation as realistic as possible. The best way to achieve this is to make interviews with domain experts. It could be that decision rules that are sometimes inside the heads of the domain experts but are not recognized by them as such. Also for knowledge, knowledge out of literature should be used. For the creation of rules for recognizing gases by the crisis center for instance, we made use of some real complaints made by the general public at DCMR about certain gases [Kui06]. For information about the levels of danger of certain gases we used the reference guide by [Rui00] which contains the dangerous concentration levels of over 2000 toxic gases. For each gas is indicated at which concentrations a warning should be issued or people get health problems and at which levels a gas could become lethal. Additional infor-

mation about specific characteristics of gases can be found in the computer program CAMEO, which holds a detailed database with descriptions on gas properties but also safety measures, precaution and what you should do when you come in contact with a certain chemical substance [EPA].

Chapter 3

Global Design

The design of MACSIM is discussed in this chapter, starting with a short introduction into the world of MACSIM. This means an introduction to the global concepts and their interdependencies (World Model). Also the data flow or messages that are being transmitted between the different (global) components are discussed here. From then on we zoom in to the MACSIM program and its actual different components. This will come down to a description of the global functionality of the different components. The actual implementation of these components is described in the next chapter, Detailed design.

3.1 Introduction

3.1.1 Introduction to MACSIM concepts

In Figure 3.1 a Global view or World Model of MACSIM is presented. This model was based on the interviews that we had with domain experts and was designed in such a way that it resembled the situation that is currently used, while still not too complex of structure. The main objects in the our model of the world are actors and waypoints. The relationship between those two types of objects is that actors move around in the world and observe physical data that is stored in waypoints. It shows three developing environments during simulation, each taking place in their own world, The Real World, The Observed World and the Crisis Center. The first world is called the Real World; obviously it's not the actual real world because we cannot generate a disaster at a real crisis location. Therefore we consider the Real World to be a simulation of the real world, that is, a model of the real world good enough to fit our needs. In this world there are a lot of waypoints. Waypoints are considered to be data storage points with information about the area directly surrounding a certain point. These waypoints contain a lot of data that the agents walking through the area use to gather information about the crisis at hand. Those waypoints are located all through the area. Some of those waypoints coincide with chemical detectors. The main reason we use waypoints is because of the reduction of computational complexity that is achieved when we only have to calculate gas concentrations or wind speeds for a fixed number of points in the world instead of for every possible coordinate in the entire area. As long as the computations for a certain point are reasonably accurate it will serve our purpose very well.

The second world is called the Observed world, because it is in this World all the agents see the events unfold and report about it to the Crisis Center or to other agents. In Figure 3.1 we can see that the different agents can report things to each other. Also they can complain or report things to the crisis center [KBR05b].

The chemical expert from DCMR has a car equipped with devices at his disposal that can specifically measure chemical components on any given spot. It gives information about the interpretation of the measured data. This information is being made available to other agents. Initially the chemical experts are located at the DCMR headquarters, and only if the crisis center finds it necessary to call them into

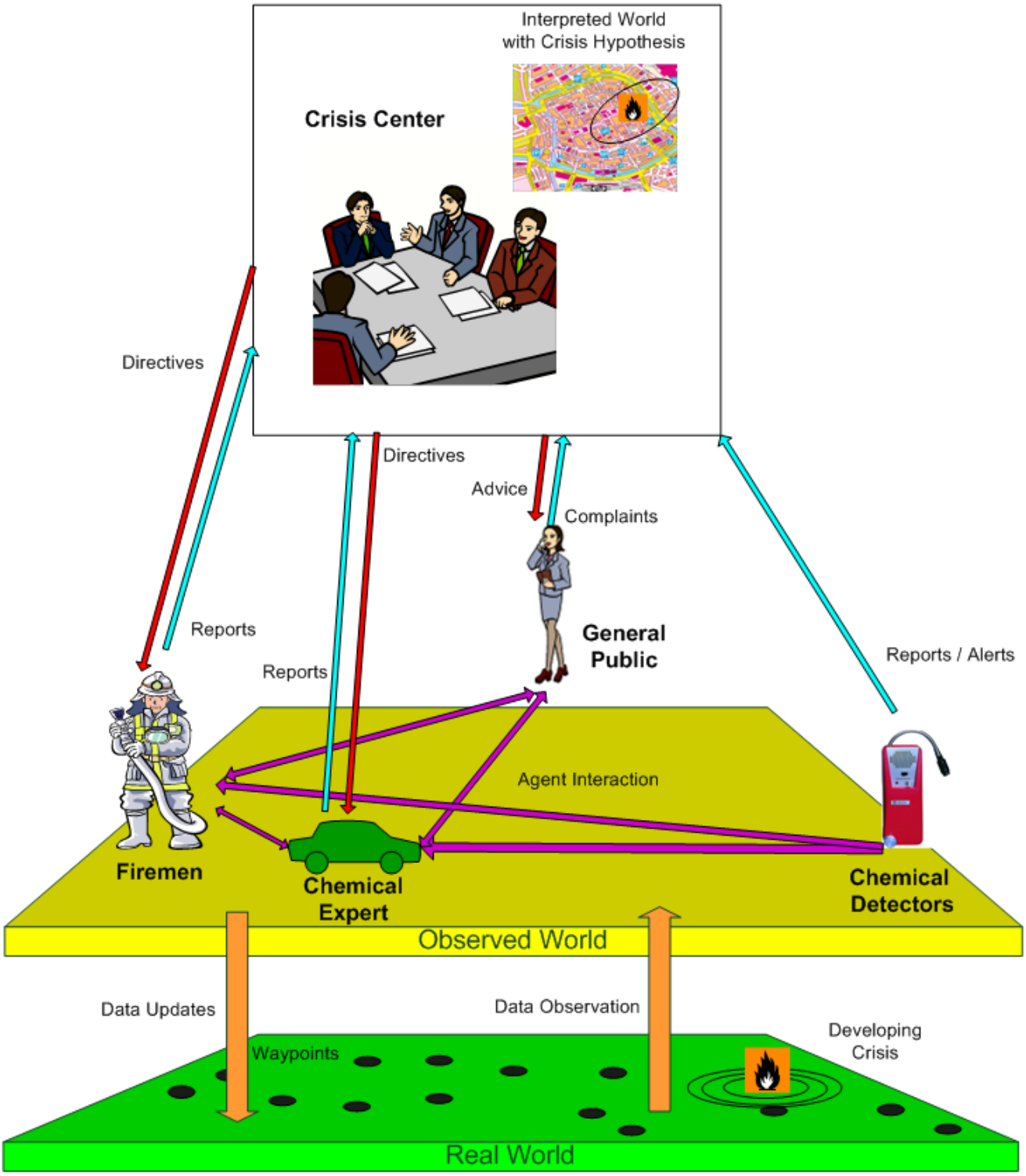


Figure 3.1: Introduction MACSIM Concepts

action, they come into play and go to a potential crisis location. From there they report to the crisis center. They can also interpret chemical detectors [vHBR06].

These fixed "smart" chemical detection devices are placed on certain locations in the world. These machine agents are able to send information if necessary and will also be able to report to humans or to the Crisis Center.

The general public just walks around randomly and runs away if something dangerous happens. They call to inform the crisis center about what's happening. If they smell something suspicious or if they see something odd they give that information as a complaint. The general public can give information to experts such as Chemical Experts and Firemen when they are at the same location. Firemen can give information about fires but can also extinguish fires if necessary. They initially are located at the fire station, but if necessary they will on their way to report to the crisis center about a potential crisis of just to fight the fire [SLOdS05], [KBR05a].

All information of the observed world is being sent to the Crisis Center. The Crisis Center will be able to either inform the agents in the world about what's going on or it will send directives. It is also able to give the most probable crisis situation and location. This is based on the fact that the crisis center has to determine what is going on in the real world based on the views that the agents in the observed world have of it. It has to make a reconstruction of the facts given by the observing agents. Based on that they come up with a crisis hypothesis of what might be going on [KBR05b].

If they have a theory about the crisis, the Crisis Center can decide whether or not action should be taken. This means that they can request the Firemen or a Chemical expert to check out a current location about what is going on. If the investigations of those expert agents give enough evidence that there is a problem, they could order further units to go to the site and aid in the crisis response effort. Also advice to the general public is being given about the situation.

It should be stressed however, that the agent types described here are not the only ones that can be part of the MACSIM environment. It is very well possible that later agents such as police officers or medical teams or even terrorists or bomb experts will be added. They of course will have their own set of properties and behaviors, which will enable them to participate in scenarios. At this point they are modeled but not yet implemented, although in next implementations this will certainly be the case.

3.1.2 Global View on Communication

The data flow between different components of MACSIM can be organized in a view as shown in Figure 3.2. In this view we can distinguish a Simulation Layer, Agent Middleware, an agent layer and one or more GUI's. It gives a clear overview of the flow of information and in which way it is being transferred to the components in the system.

When we look at the Simulation Layer, we notice that there is scenario storage. In this storage (it could be a database or just a file with a scenario in it) one or more scenarios are being kept. A scenario can be acquired from the storage by the simulator. The simulator is in charge of simulating crises and for that it needs scenarios. Those scenarios are being transformed into a script internally by the simulator. A script is basically a timeline with a start time and an end time and certain events that can take place in the world in between. The simulator is processing those events and this usually means that as a result of a certain event the world is modified in one way or another. Therefore the simulator is updating the world. When the world is being updated, the agents that move around in the world should be notified of this, because they have to sense and experience those changes in the simulation.

This is when the agent middleware comes into play. The agent middleware takes care that the agents in the simulation are receiving the updates of the world. The reason that agents are receiving this

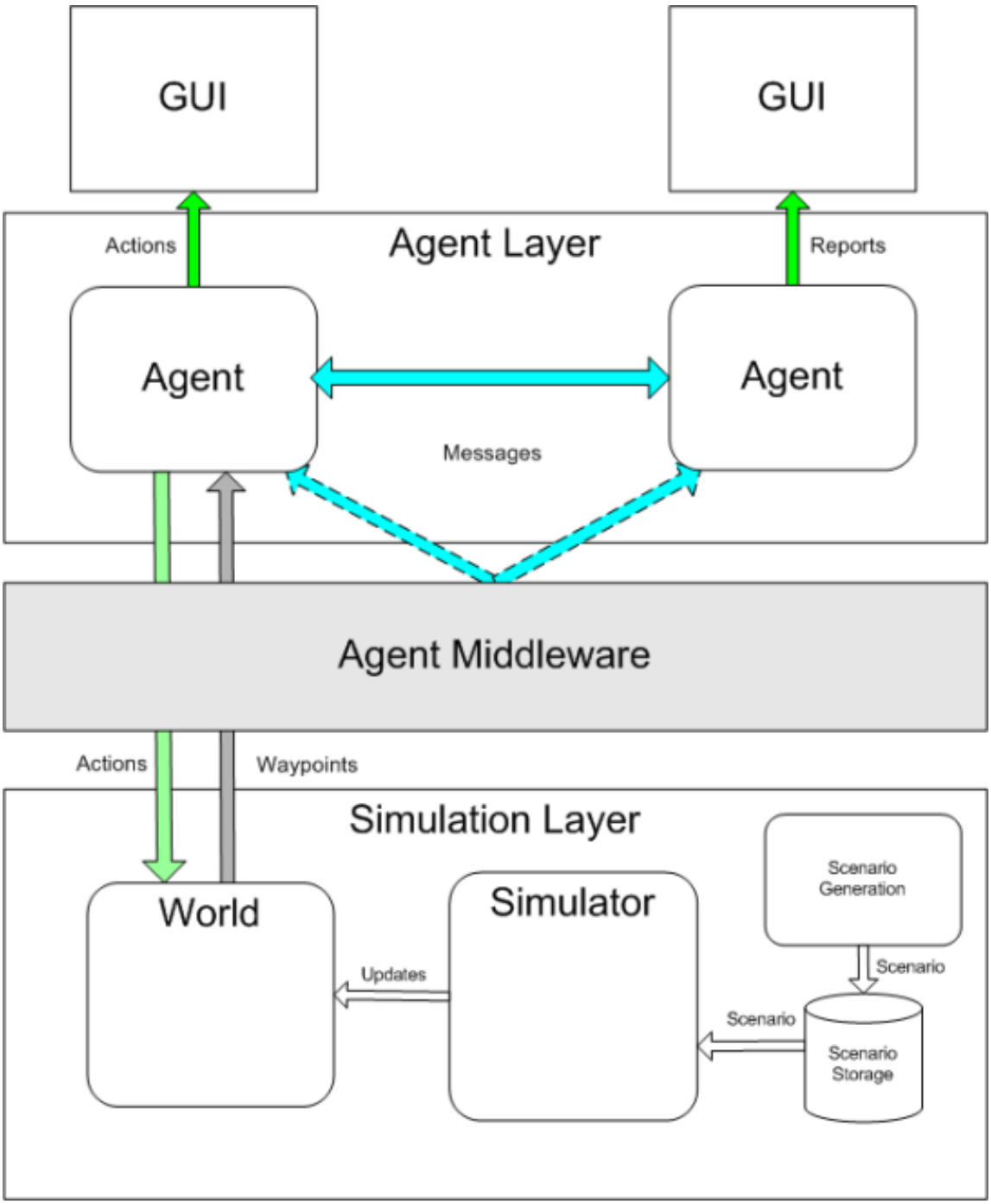


Figure 3.2: MACSIM Global Data Flow

information via agent middleware instead of directly is based on the fact that agents are supposed to be autonomous. This means that the agents are supposed to work as independently as possible. Therefore other components should not have direct access to the agents, because that would imply some sort of ownership that is incompatible with the idea of independent agents.

In the meantime, the agents are receiving updates of the world in the form of information that is stored inside waypoints. Those waypoints, as we have seen in the concept in the introduction, contain all different kinds of environment data that can be read by the agents. If these agents sense this data they can process it and then reason about it. Based on this reasoning the agents are initiating actions. Those actions might have an effect on the world or not, but this is of course depending on the type of action that is the result of the agent's reasoning. Besides reading waypoint data, the agents are also capable of sending messages to other agents. In the diagram the blue arrow indicates this, but it would be more accurate to connect the arrows via the agent middleware. This is because of the same reasons of agent independency. Those messages are being sent to other agents through the agent middleware as well.

The agent actions that have an effect on the world are being propagated back again to the world to implement the changes inside the world. This of course requires some sort of synchronicity scheme that makes sure that the simulator applying the script to the world knows about the updates by agents, so it can update the world again accordingly. The synchronicity scheme chosen for MACSIM will be discussed in greater detail later on.

Finally the users of MACSIM will be able to view agent actions. This means that the agents will have to have some sort of GUI because otherwise the user will not notice their actions. If they have received or sent a report, then its GUI must also be showing those, for information purposes.

3.2 Design of the System

The components of MACSIM will now be discussed in more focus. The system is a first prototype for a crisis simulator as discussed by Dr. Rothkrantz in his paper about serious gaming [Rot06]. It is important to stress that the creation of this software will be a permanent work in progress. Because the components of the system are diverse in number and function, it leaves a lot of room for expansion and improvement in the future.

3.2.1 Goals

MACSIM should be:

1. **Expandable**: Because the system being a research object, it should be able to incorporate new features that could improve the functionality of the system but are currently not yet in the scope of the system. In the design we should take care that we design as 'open' as possible, to facilitate future work on the system. We therefore chose an incremental approach, by means of implementing prototypes in which each time additional functionality is implemented.

2. **Not memory consuming**: A system containing a complex gas dispersion simulation and other physical properties for thousands or millions of points in the simulation world should be programmed in such a way that not too much data traffic is involved and data retrieval is efficient. The updating of values inside a physical model can, when millions of points are involved, very time consuming. The agents that are at a certain location must get the data they need as fast as possible, so their observing and interpreting of the data does not get delayed.

3. **Easy to modify with respect to constants**: Being a simulation, the system relies on a lot of mathematical, physical and empirical constants. It should be avoided to include those constants in Java code, because this would mean that if they should be changed for some reason, that people

have to dig in the Java source to change whatever they want. Changing those constants should be very easy to do.

4. **Operating on JADE**: With JADE being one of the leading platforms for agent programming, it was decided that the simulation should run on JADE, to facilitate running of parts of the system on mobile devices in the future.

5. **Incorporating Real Knowledge**: For making a simulation that is believable, it is important to include real or at least realistic data into the system. This means knowledge from experts and existing data from crises from the past, and real physical data from the real world, to esnure that the simulation does not become some sort of fantasy game.

6. **Facilitating Interactive Simulation**: The future of MACSIM will be in the area of serious gaming, so therefore it is important that we make sure that the design is fit to accommodate the features required for the implementation of serious gaming. This means that the users should be able to feel presence in the future implementations of MACSIM and are able to get a good training experience out of a session with MACSIM.

The nature of the system is a proof of concept for the ideas for serious gaming and simulation in the field of crisis response. In the system we try to show several aspects of crisis management, over an environment that enables users to program simulations and different events. Every design decision has to be made with the intention that it has to be easy to add functionality to the code later on.

In the design as it is right now, several components are made. During this project most of these components will be partly implemented, to provide a basic form of functionality, to see whether we are in the right direction. The goal of MACSIM is to produce a broad-ranged prototype of a simulator that can be used as a demonstrator to illustrate the ideas for an agent-based crisis simulator.

3.2.2 Components

Here the different components of the crisis simulator will be described. The actual technical details will be described in the corresponding chapter of this thesis (Detailed Design), but in Figure 3.3 a global description of each proposed component is given.

Simulator: This is the main component of the program, where the simulation is being set up and run.

World: the simplification of the world for which we are making simulations. We cannot incorporate everything that is in the real world into our world model. Because of complexity and performance reasons we make the world in which a scenario is developing as simple as possible.

Physical Model: The collection of models that enable the events to impose changes inside the world. These are mainly a set of formulae that can be changed or added when necessary.

Event Generator: This component makes sure that events, that are part of a scenario, are executed in the right way. This means that the event generator takes care of updating the world model in the fashion that a particular event requires.

Simulation Setup: In this window all the properties of a certain simulation can be edited, loaded and saved. After the user is satisfied with the scenario, it can be stored and executed.

Agent Middleware: The agent middleware ensures that the messages that are sent between agents are being processed correctly. Also the updates of the world are processed so that the agents have them at their disposal.

Participating Agents: These are the key elements in the system. They are the main characters in the

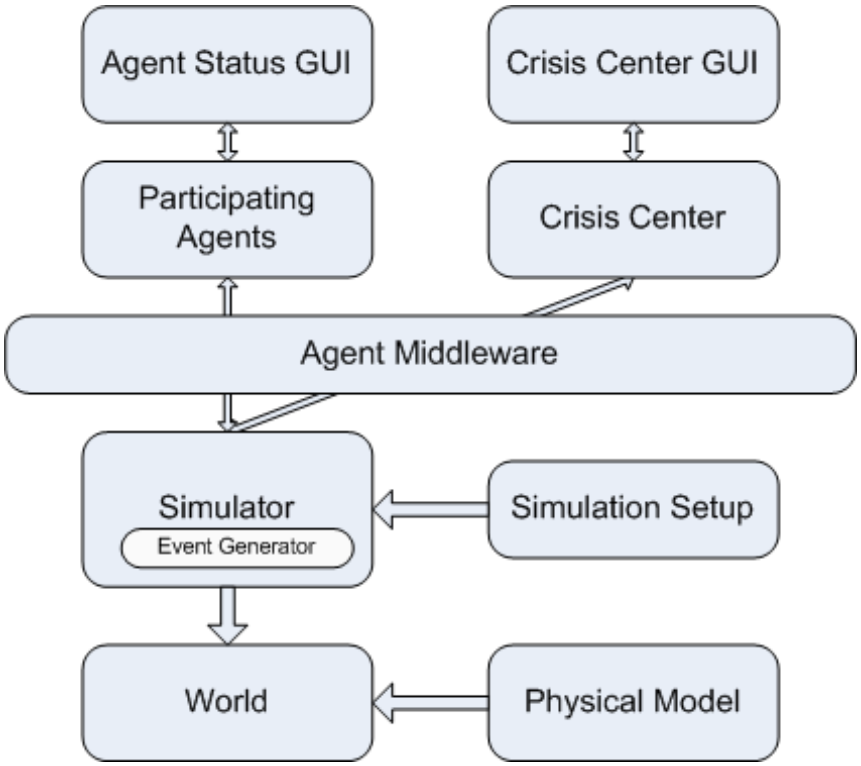


Figure 3.3: Global Component Overview

simulation and as the scenario unfolds, the characters respond to the changing situation. The agents can communicate with each other about the current status, and they can use reasoning for making decisions about what they should do the next. Participating agents can also contact the crisis center.

Crisis Center: during a crisis, a crisis center is usually being set up to coordinate the crisis response effort. Here all the information about the crisis comes in and is being interpreted by the people who are there. They issue commands to the participating agents that are in the field, based on the information that they get from all available sources. This information is being stored inside something that is called an interpreted world. This is an interpretation of what the crisis center believes is going on inside the world. In real life, the interpreted world is usually some sort of real or digital map onto which status information is being added. Based on what they see on the map, the crisis team decides what to do.

Interface during simulation: During the simulation the agents are making all kinds of decisions, and a lot of communication is exchanged as well. We want to keep track during run-time of what is going on in the world, so we need some way to eavesdrop on the communication during a crisis. In quite the same way that communication during a flight is being recorded, the agent communication will be shown on a user interface while the simulation is running.

3.2.3 Program Flow

In the final program most of the components will be shown in a somewhat simplified form, but with enough functionality to give a proof of concept for an agent based simulation of a crisis with added AI functionality. Figure 3.4 gives some idea of what the user will see during the running of the prototype version of MACSIM.

As the system is started, a scenario is being loaded. At this point the user is able to edit the scenario and set some parameters of the scenario, such as the start time, end time, map size etc. Then the simulation is

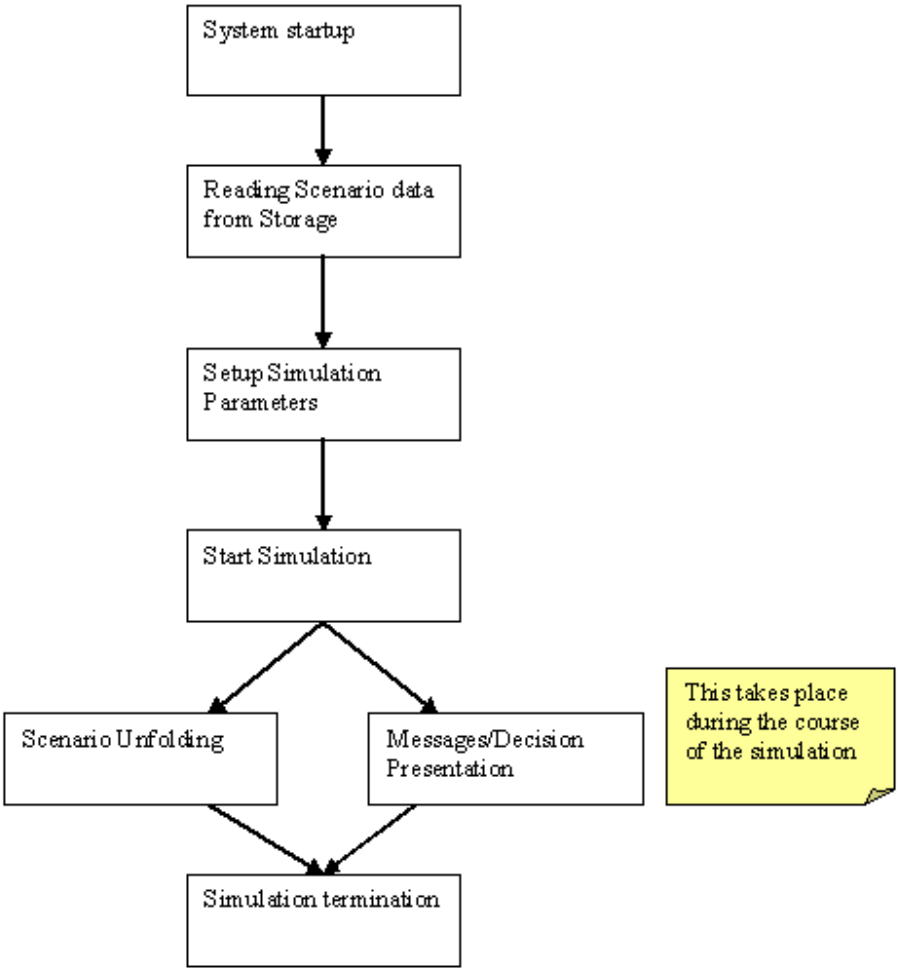


Figure 3.4: MACSIM Program Flow

started. When the scenario unfolds, more and more messages start to come in and the Crisis Center will present a hypothesis of what is going on. Eventually the simulation will stop and the user can terminate the simulator.

3.3 World Model Assumptions

In this section the world assumptions that are being made for the first version of MACSIM are being described, as is shown in Table 3.1. It must be said however that some of the assumptions made here are only temporary assumptions that will be dropped in later versions. Because for producing a first prototype and therefore a proof of concept full functionality is not required, and therefore the world model at his point only contains the bare necessities. This means that information about wind direction, sounds that can be heard, gas, fire and smoke information is currently included. Some information at this point is still left out, but in the following sections we will explain why we currently made some assumptions to leave some data out of the system at this point. It should be stressed here that these assumption are only temporary. They are not assumptions that are permanent in any sense and can always be adapted if necessary. Later it could be decided to use a game engine to represent more elaborate data in the world to represent effects of crises. That would enable us to drop a lot of the assumptions made in the following paragraphs.

We consider the world to be a place where chemical substances can escape, fires can break out, explosions and flashes can be seen and several agents can move around. Buildings are not being modeled for the time being, because it makes the simulation of gas dispersion too complex. In real life, gases move around a building. This makes the local concentration a little bit higher or lower than when the concentration is measured in a place where there are no buildings. This assumption can be considered a valid assumption, because the most important thing is not the absolute accuracy of the gas model used, but the proof of concept of using a certain gas model in combination with a real-time AI reasoning engine. For this engine it's not absolutely necessary to receive 100% accurate data, but the fact that it is data with a reasonable level of accuracy combined with the validity of reasoning rules should be enough assurance for a credible simulation of the real data interpretation.

Table 3.1: MACSIM World Model Assumptions

World Model Assumptions
Gas Information
Fire inforamation
Explosion information
Atmospheric properties
No Buildings modeled
No Height Differences
No Roads
Waypoints for Computation Reduction
Some Quantities constant for entire World

Another assumption in the model of the world is the absence of height differences. There will be no hills or small fluctuations in the landscape. The reason for this is the same as the reason for the absence of buildings; it was a tradeoff between computational complexity and the need for data with such a high level of accuracy. In practice, it was an allowable assumption, because in the Dutch situation the difference in height in most cases is negligible.

Roads are also purposely not included, because of the addition of complexity that it brings while using persons in the field that are moving independently. It is very restrictive and algorithmically complex to only allow agents to move via indicated roads. We assume that agents move around, but where they are doing that and how is not our concern. They can move freely and without any restriction through the world. Of course we need to make sure that they cannot walk through situations, which cannot be trespassed anyway, such as fires, or water.

To prevent that we have to calculate the physical situation for every square millimeter in the simulation area, which would computationally far too complex, we use waypoints as data structures. The waypoints contain the physical data for a certain square area surrounding it. This means that for a certain location (x, y) we only have to retrieve the most recent waypoint. This assumption is of course only valid if the number of waypoints in the area is not too low. On the other hand, if we have too many waypoints in the world, the gain in computational complexity will be reduced significantly. To achieve a reasonably accurate simulation result, we have to find a right balance in the density of waypoints.

For now, we also assume that some quantities in the world are the same for all waypoints. This is mainly because of time limitations. Quantities such as wind speed, temperature and wind direction are being set as a default value. Models to describe those values are of course available, but the question remains whether these models are not too complex to serve the proof on concept well. In the Physical model there always remains the possibility to add formulae for wind direction variation, wind speed variation or temperature differences.

3.4 Simulation

3.4.1 Script Execution

One of the key components in MACSIM is the Simulator. The Simulator is in charge of regulating the simulation in such a way that every time-instance the necessary updates to the world are being made. These updates are initiated by a script that has time-based events listed. For controlling the script a special component called a script controller is defined. The script controller manages things called event generators for processing the events inside the script. Those event generators are in charge of letting an event unfold in such a way that in every time instance the right parts of the world are updated.

The world on which the updates are performed consists of waypoints. Any information about the world is contained inside the waypoints. This gives the freedom to implement everything that is to be "sensed" in the waypoints. So besides climatologic and sensory values suggested in the World Model we could also model other things (water, for instance). This means that not exactly a world is updated, but more a set of waypoints. This set of waypoints could be all the waypoints in the world or just a selection. This is up to the event generator that is updating the world. An update of a waypoint basically means that one or more values inside a waypoint are replaced by a new value.

There is a difference between scripts that are just simply executed by the simulator and scripts that allow agents because of their responsive action to be changed in course and order of execution. The first one is called static scripting, the latter one dynamic scripting. At this point, in the design of MACSIM,

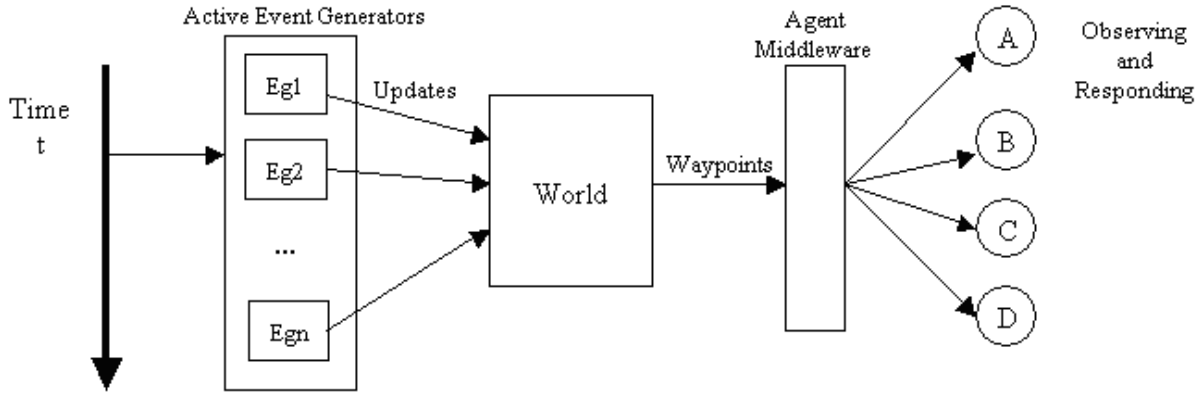


Figure 3.5: Update Algorithm Concept

according to the design goals it should be clear that in theory both types of scripting should be allowed. The only problem is that the actual implementation of dynamically adding of events has some practical outcomes that have far reaching consequences.

At this point, the dynamical addition of script events is not implemented, because it was not required for the proof of concept. It has to be emphasized however, that adding events dynamically most certainly is part of the design. For dynamic scripting the agents need to have influence on the development of the simulation that is being run. This is very difficult because every action of agents in the field can change the overall development of the scenario and therefore can change the course of the simulation. This brings forward some important questions about the concept of a scenario, for instance to which extent a scenario can be implemented (ie read from storage) and to which extent it can be defined in real-time. This was considered not essential functionality for the time being, but it will be top priority in future implementations.

3.4.2 Waypoints and their Values

The waypoints could contain all sorts of values. The agents in the area surrounding that particular waypoint can observe these values and interpret them in whatever way they fancy.

Some examples of events that can be sensed are:

- Visibility – It could be that the area is very foggy and therefore agents cannot see anything in the direct environment. This could lead to changes in behavior of agents in their moving patterns or emotional state.

- Temperature – The temperature can for some agents be a reason to get away from a place, and for other agents reason to get closer, for instance a fireman.

- Wind Speed – When a gas is released it is important to know for an agent what the wind speed is. A chemical expert can give a warning based on it, because the gas will of course spread faster with a strong wind.

- Gas Information – Specific information of gases that are spread into the world are being put in one dedicated data structure, this can have the advantage that whenever in a later state new gas data is required it can be added to all gases at the same time and all gas-related data is located in one place. For now you can think of several quantities as danger, color, gas concentration, gas behavior, medical hazards and smell. For different agent these could create different reactions.

- Sound of a Bang – An agent can hear a bang or a lot of noise. This is of course related to an explosion or another event that creates a lot of noise. Depending on the cause of the sound is heard for a long or a short time.
- Flash – A flash of light can be caused by a huge fire or lightning or a huge explosion. Depending on the type of incident that the huge level of light is causing, is could stay visible for longer or shorter time.
- Level of Fire and smoke – Based on the severity of the fire these level are raised when a fire breaks out.

Obviously not all agents can observe the same values. Some values are only interesting or noticeable for specific agents or at a special time and location. Therefore we need some sort of scheme to prevent certain agents to see certain values in a waypoint and to allow other values to be seen. It will come down to a list of value types for each different type of agent in which is listed what type of data is "visible". This is of course depending on the observing skills of the agent.

3.5 Physical Model

At this point there are only a couple of events available that can be launched from the script. Here only the design rationale and practical considerations of those models for the MACSIM design will be discussed. The implementation details of these events will be discussed in the chapter about detailed design.

3.5.1 Gas Dispersion Model

For a simulation with gas escape involved, we need to have a gas dispersion model at hand. Therefore a lot of effort was put in finding a gas dispersion model that was simple to implement and yet accurate enough to represent a real gas dispersion. There are several alternatives for gas dispersion models available commercially, but the problem with those models is that they are far too advanced and complicated to include in a simple software simulation. For instance, the SAFER gas dispersion model is the standard software tool for the chemical industry to model gas dispersions. Besides that, the real-time gas simulation model included was obviously copyrighted and could not be implemented (read: reverse-engineered) that easily. It works in real-time and has a fantastic graphical interface and a lot of parameters that make it a good tool for the chemical experts, but not really a good model to implement as a basic gas model in an AI simulation. It might become a good reference in developing a user friendly simulation environment.

The other easily available gas dispersion model was ALOHA [EN04]. This easy-to-use gas (and liquids) dispersion model can be used for free for educational purposes. The plotting of the gas dispersion on top of a map of a certain area and a reasonable approximation of the actual gas concentration on a given point (x, y, z) in a time extent of an hour were two interesting features that will be part of the MACSIM requirements.

It also gives you the possibility to draw isocontour graphs on top of the map to indicate in what region dangerous concentrations of that particular chemical substance are being measured. The only part that wasn't available in ALOHA was the real-time element that we want. It can only give a plot of concentration against time for a certain location, but you do not see the actual development of the gas cloud, as was the case with SAFER [SAF]. Besides that, only maps of American counties were available, so we could not use this model either, but it came close to what was needed.

Although ALOHA was close to what was needed, it still was a finished application and despite being free for academic use, still a bit of a black-box. Therefore it still was difficult to derive the models that

were behind this. Fortunately, on the homepage of ALOHA some basic information on the models is given. ALOHA uses two gas dispersion models, one for neutral/light gases (gases lighter or as heavy as air) and one for heavy gases. The model for light and neutral gases is a Gaussian-Plume-Model (GPM). For heavy gases the DEGADIS model is used, that takes into account that heavy gases first sink to the ground after which they still start to disperse in a Gaussian fashion, only a lot slower than lighter gases [CPR97].

A well-known example of a Gaussian model that is open to the public is the one that is described in a publication of the Dutch Government called CPR14E, with the title "Methods of the calculation of Physical Effects" or the "yellow book". Currently, the 3rd edition is being used [CPR97], for simplicity reasons we used the GPM mentioned in the 2nd edition [CPR92]. This version of the model uses far less complicated formulas and is more easy to use in different weather circumstances.

The GPM model is a generalized gas dispersion model that is applicable for light and neutral gases. For heavy gases it is only usable for long distance concentrations as heavy gas first sinks to the surface before starting to disperse in a somewhat Gaussian fashion. At this point there is no model included in MACSIM for short-distance effects of heavy gas dispersion. But the way the physical model is designed it can always be added in later.

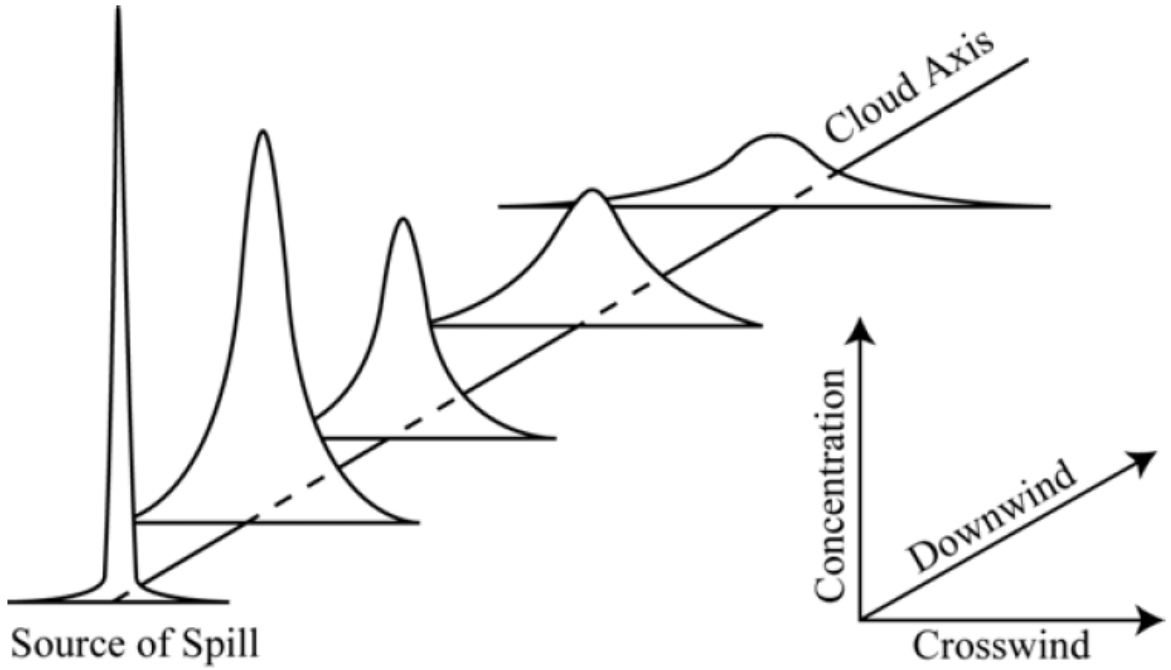


Figure 3.6: Gaussian Plume Model [CPR97]

The most important gas dispersion assumptions for the world in our simulation are the following [EN04], [CPR97]:

1. No gas is lost after dispersion.
2. The area must be flat and open.
3. No obstacles can be taken into account.
4. The meteorological conditions are constant over the entire area

5. The measured concentration is an averaged value (mostly because of turbulence in the air concentration may fluctuate from time to time).
6. There must be some wind present (wind speed of at least 1 m/s).
7. The dispersion source has no dimensions (this is to make the calculations less complex).
8. The dispersion source is located at ground level.

Every gas that is known inside MACSIM will be dispersed with the GPM model. This is mainly because the heavy gas models require a lot of physical calculation including thermodynamics and gas flow calculations, because the gas first has to sink to the ground and from then on disperse in a way that light gases are dispersing [CPR97].

3.5.2 Fire

The fire model inside MACSIM must be able to spread from the origin to another place. Based on the type of fire it should expand in all directions. This also implies that there should be different types of fire. At this moment we have accounted for two different types of fire, namely a small fire and a heavy fire. This requires an algorithm that is able to let a fire spread in such a way that depending on the type of fire it takes longer for a fire to expand and the fire spreads further. It should also be possible for firemen to extinguish the fire and depending on the severity of the fire it might or might not come back. This is something that might not be required for a first demo, but in the design we have to acknowledge the functionality.

3.5.3 Explosion

An explosion at this point should at least generate a flash and a bang. Because of the world model assumptions that we made, other effects that you could possibly model, like damage on buildings or infrastructure are not available, because of the lack of buildings and infrastructure in the current model. But should there be a new world model this would be the first effect of an explosion to be implemented, because it adds the essential part of realism to an explosion. When for instance a game engine is being used to model additional features in the world that would create a 3D atmosphere, and would expand the possibilities of modeling the effects of explosions because with environments like this effects obviously can be easy visualized.

3.6 Multi-Agent System Design

3.6.1 Multi Agent System

If MACSIM is supposed to be a simulation in which multiple agents observe physical properties in the environment and respond to it, we have to make sure that the design of a Multi-Agent System is well defined and serves the purpose of the system. As JADE is going to be the platform or agent middleware on which the system is going to run, the agents have to be JADE compatible, but furthermore, the simulation has to be JADE compatible as well. The agents in the world are supposed to be autonomous or independent in their acting, but the updates of the world in the waypoints that are in the vicinity of the agent still need to be delivered to the agents. The data updates are applied to the waypoints whether an agent is in the neighbourhood or not. This creates a potential conflict because for updating the data that an agent should own the simulator should know the agents and thus should update their parameters, but this is in contrast with the independence and autonomy of an agent. It is even true that an agent, like

a real human being, at the moment he opens his eyes is observing the world. It is therefore always autonomous as he has the freedom to do something with the information that is presendted to him or not. In either case we have to devise a scheme that enables us to indirectly update the waypoint data of an agent.

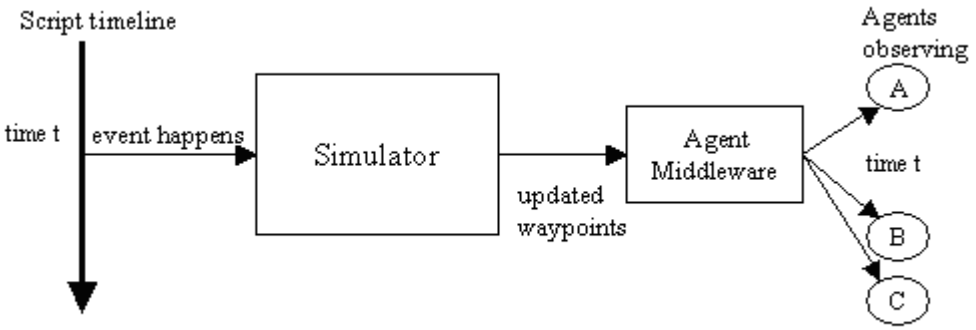


Figure 3.7: Synchronized Data-Observing

Another requirement is that everything the agent does should be performed synchronized with the actions of the other agents. This is because the simulation is script-based. This means that when an event E takes place on a time-instance X the waypoint data updates that are the result of this event E must be read by all the agents on the same moment. Otherwise the agents would be reading waypoint updates from different time instances simultaneously. That could mean that different agents in the same simulation are in a different time instance, which of course would be bad for the level of realism in the simulation. Therefore the execution of actions by agents and the reception of waypoint updates should be synchronized.

3.6.2 Participating Agents

Currently there are four types of participating agents defined in MACSIM: the General Public, the Fireman, DCMR Expert and the Chemical Detector. These agents have their own function in the world. While the simulation is going on, they observe the values that are stored in the waypoints. They process them based on their capabilities. Besides these participating agents, there is also another agent that partakes in the simulation. This is of course the Crisis Center. It receives the updates of the world indirectly through information from the outside world. In the following section we globally describe for all these agents a general set of characteristics and what they can do inside the world.

General Public

Members of the General Public are non-professionals in crisis management that walk around in the world and act as innocent bystanders who are affected by an incident or just observe something strange but do not know any specific details about it. They can feel responsible to report about events which in their eyes might seen anomalous. They can be afraid or they can inform through their mobile phone. They walk randomly and complain or inform to the crisis center about what they are experiencing. Something could smell, they could report a certain color in the sky, a bang that they heard or a flash that they've seen. Other types of agents can report this as well, but for the general public this is the only type of reaction that they can give.

Fireman

A fireman can give the same information as the general public, but when it comes to fires they of course know more about that and give more specific details about such events. Detailed information about fire levels, hazards, smoke levels etc. can be sent. Of course the Firemen can also extinguish fires. They are not on the location of the incident when the incident starts. They are idle at the fire station. When they get orders from the crisis center to go to a certain location they will go to the location mentioned in the report. Based on their expertise they will investigate what is going on and report back to the crisis center. Of course a fireman is wearing protective clothing, so he is not too bothered by fire or smoke. Because he has a fire engine the Fireman can drive to a location very fast but he can also walk around to extinguish a fire in another waypoint if necessary.

DCMRExpert

The DCMRExpert is a chemical expert who is able to drive to a certain area and take exact measurements of chemical substances in the air. Just like the Fireman, he starts at the DCMR office where he gets a Report from the CrisisCenter about what is going on. He then drives to a certain point. After putting on his protective suit he takes measurements and makes a Report about it for the CrisisCenter. While taking measurements, the DCMRExpert agent is able to determine what risk a certain concentration of a chemical means for the area, and he knows everything there is to know about a chemical incident, where the Fireman's knowledge is a bit more limited.

Chemical Detector

The chemical detector (for now) only sends reports about certain gases that are measured by those systems. These agents are emulating the real chemical detectors as good as possible. They are sending Reports to the Crisis Center every couple of minutes about the concentrations of gases that they are supposed to be monitoring. Of course if there are concentrations that are alarming, the chemical detector must report about that as well. The chemical detectors are placed at certain predefined locations and cannot move around.

Crisis Center

A crisis center is an office where the "powers that be" confer after an incident has taken place. They get information from people in the field who report about what they sense or experience. Based on this input, the Crisis center determines what the reaction should be. The crisis center can for instance send the DCMRExpert or the Fireman to a certain location if they think that is necessary. Furthermore it is the job of the CrisisCenter to combine data that is coming in intelligently to give an adequate response to the current situation. They can also give the order to the general public to leave the area. They keep track of all the information that is coming in inside their own interpretation of the observations in the world, the Interpreted World. Besides that they have a graphical representation of the world, a crisis map, which can indicate updates of events that happened on top of it.

3.6.3 Communication

Agent behavior and data traffic is just the basic layer for implementing real Crisis Response communication. The next step will be to implement communication mechanisms to make the simulation realistic and to actually simulate communication by crisis management agencies in situations of an incident. Obviously the communication in the system takes place through messages. These messages can be exchanged by different agents. We assume that these messages are in real life being sent by mobile phones or the C2000 system [KBR05a] or by regular speech. We will not model these different ways of sending messages individually but just assume that the messages come in via the means of communication that

the agents will most likely have at their disposal during the course of the simulation. The messages themselves will be sent via JADE middleware in the actual code.

There will be different types of messages in the system. At this point we make the distinction between messages that are giving information and messages that are used for giving orders. We call these messages Reports and Directives respectively.

Reports

These are the packages of information that are being sent to communicate between the agents in the field and the Crisis Center. In these packages should at least be mentioned: who sent it, what is going on, and where. In these messages an entire range of different messages can be sent. For the different concepts we use the ontology that Siska Fitranie created [FR06], which can be easily converted to a Java, Jess and JADE compatible set of data structures. We will try to utilize the structures currently available in the ontology. This leads to this ontology being an interesting testcase to test the usefulness of this ontology in crisis messaging.

Directives

Directives are messages simply containing an order or a request to do something. It contains an order and a location. Examples of orders are "go to" and "retreat" in which the location mentioned in the message either means that the particular agent should go there or should get away there. Because of the proof-of-concept nature of the system, it could be that the structures of both messages types will change if they do not fit the purpose of the message.

3.6.4 Reasoning

When an agent has received an update about the current situation, he must be able to respond to it. He does this by reasoning about the information that is coming in. Based on the reasoning based on the mental model he decides to do something. This is the so-called model-based reflex-agent paradigm of [RN95]. For the crisis center, because it has a model of the world that is internally stored based on the input of messages and possible additional a-priori knowledge. The main goal for making decisions rules is to give the agent an idea what is going on in the area of interest. For the participating agents, this will mean what's going on at his current position, while the Crisis Center wants to know what the global situation is in the world as a whole.

After a decision has been made about what is going on, all agents will communicate with other agents about this situation, the participating agents could discuss in what kind of way they want to should solve the crisis. The Crisis Center sends orders to the agents in the field about what they should do based on the reasoning. So apparently all agents use reasoning for different purposes. Therefore the type of rules they use and the messages they send will also be different. The rules should be implemented inside the Jess environment.

Participating Agent Reasoning

The reasoning design of a participating agent is simple. It only has to be able to process sensory input and interpret it in its own characterizing fashion and after that report about it in the correct fashion to the crisis center or initiate an action based on it. In this version an object representation of a complaint

will be sent to the crisis center. Nevertheless a participating agent will have to construct the object representation of a message based on decision rules. So each participating agent will have to create a set of rules that enable him to create message objects and execute agent actions based on its sensory input. This scheme is visualized in Figure 3.8.

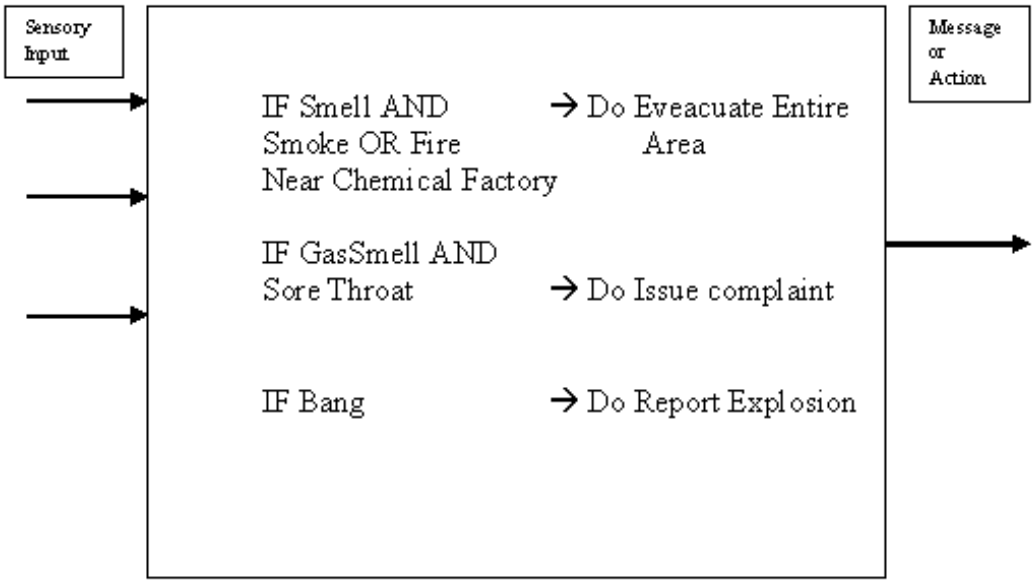


Figure 3.8: Sensory-based Rule Firing

Crisis Center Reasoning

The input that comes in at the crisis center is also being compared based on rules in a rule base, but there is also a model behind this with a-priori knowledge that the crisis center has available. This information is stored inside something called the interpreted world. It is a representation of the world, as it is known inside the crisis center. Also the crisis center must be able to determine, based on the data that is coming in from the outside world, a possible situation that might be going on outside. This is called a crisis hypothesis. Both concepts will be explained here.

Interpreted World

The interpreted world is a collection of all the knowledge that the crisis center has available for getting a clear picture of the crisis situation at hand. It contains a collection of messages on which he can base its hypothesis, but there could also be a place for other sources of data, for instance geographic information or the location of agents in the field where crisis response agents are currently working. This model of the real word inside the crisis center could enhance the possibility to create smarter rules for determining what is going on and smarter positioning of crisis response equipment.

The design of the interpreted world should be made in such a way that different types of information can be added later on as layers, like layers on a digital map. In this way the Interpreted World should become more and more equivalent to a datastructure that can be visualized like a digital map. In this way we have a crisis map like it's used inside a real crisis center, and at the same time we have a source of data that can be intelligently combined to create decision rules for the crisis center. So the Interpreted world is a map and knowledge base at the same time.

Crisis Hypothesis Creation

In a crisiscenter work people with a lot of experience on recognizing different types of crisis situations based on the messages they get from the general public. If a knowledge base should be modeled after a real crisis center, we have to make a crisis center design that is a realistic approximation to the reasoning that is going on inside the minds of employees of a real crisis center. Based on the information from the general public that comes in, the people at the crisis center are using the messages to rule out as much possibilities as they can to get an accurate hypothesis of the situation at hand fast. Furthermore they want to know where the situation is going on, so they can send DCMRExperts or Fireman Agents to the most probable location.

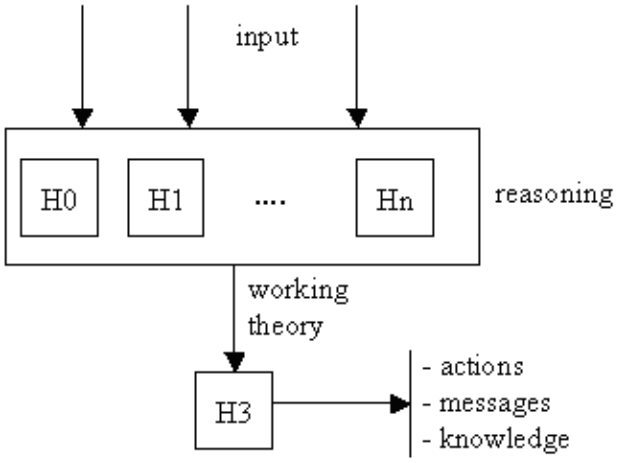


Figure 3.9: Hypothesis Forming Concept

Therefore they are looking for a certain type of probable incident, from which they can start crisis management efforts. By doing this they reduce the amount of uncertainty in their minds about what is going on. This doesn't have to be the situation that is actually going on, but at least they have a working theory from which they can start. At the beginning of the crisis situation all options are still open. But as the crisis event develops some events are more probable than other ones. The Crisis Simulator's modeling of that behavior will be described in the Detailed Design Chapter.

Chapter 4

Detailed Design

In this chapter the design will be discussed more from a Java perspective. The functionality of the code is explained, and via the presentation of use cases (data creation, world updating, agent behavior) the entire system as it works now will be discussed. Some of the code is just code that means to transport the right data to the right classes, so the focus in this chapter will naturally be put on the system's more elaborate and complex algorithms and meaning of data types used in the simulation. This section should be read as an expanded version of the documentation that is delivered with the actual code. It is made as more of a global description of what the Java code does than documentation of the Java code itself. Because MACSIM consists of two distinguishable parts, a part for simulation and a Multi-Agent System, both designs will be discussed in separate parts of this chapter. First we will discuss the use cases and sequence diagrams of the simulation part of MACSIM, in the next section the use cases (or Behaviours in JADE-lingo), sequence diagrams and class design of the agents will be discussed as far as its necessary to understand the functionality of MACSIM.

4.1 System Overview in contextual Level

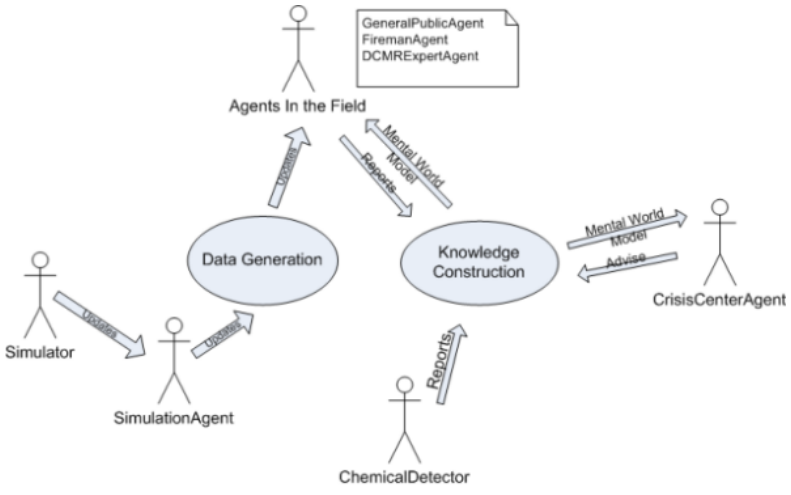


Figure 4.1: Actor Identification with main Tasks and Processes

In Figure 4.1 the actors of the system are being identified. The Simulator makes sure the world is being updated. Besides the Simulator, there are only actors that coincide with JADE agents, and they communicate with each other accordingly. These agents exchange messages of a certain kind with each other and while doing so, they are executing use cases. In the next section we will for each actor define the appropriate use cases.

4.1.1 Actors & Use Cases

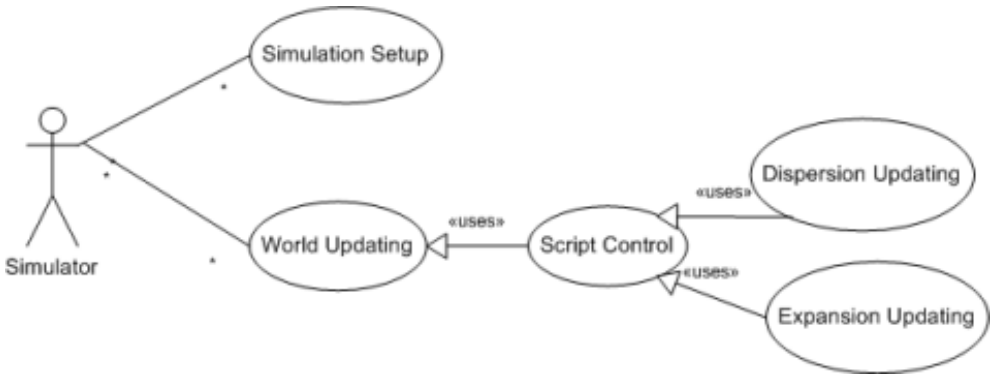


Figure 4.2: Use Cases Simulator

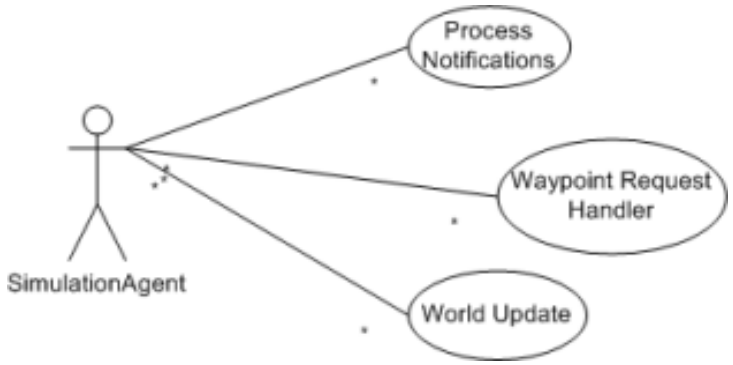


Figure 4.3: Use Cases Simulation Agent

In Figures 4.2 to 4.5 the different use cases of each actor are being defined. Most of these actors are JADE agents. JADE agents have a special way to model the actions that are being executed by Agents. They use a special dedicated Thread pool for each agent (which is in itself a Thread as well) to add actions that should be executed [Cai03]. In the implementation of MACSIM it will usually be the case that a use case is identical to a subclass of the abstract Behaviour class, which is a part of the JADE package. In this version of the implementation it will often happen that Behaviours are mostly the same for different agents. That is the reason why in the picture most Agent actors are pointing to the same set of use cases/behaviors. This is also why in some cases we will discuss the use cases/Behaviours per Agent, and in other cases we will discuss them per use case or Behaviour and then first discuss the generic part of the Behaviour and then discuss individual differences per actor.

4.2 Simulator Design

4.2.1 Simulation Creation

First the Simulator creates a Simulation, and through a graphical user interface an actual Simulation Object is created, then JADE is started up, then the agent controllers of all the agents in the simulation are created. Each Participating agent is given a unique agentCode (an integer) which is useful for the waypoint retrieval (this is to give all the participating agents and the CrisisCenter a unique identifier for waypoint updating) then all agents are started up, and they do what they are supposed to be doing, executing their set of Behaviours.

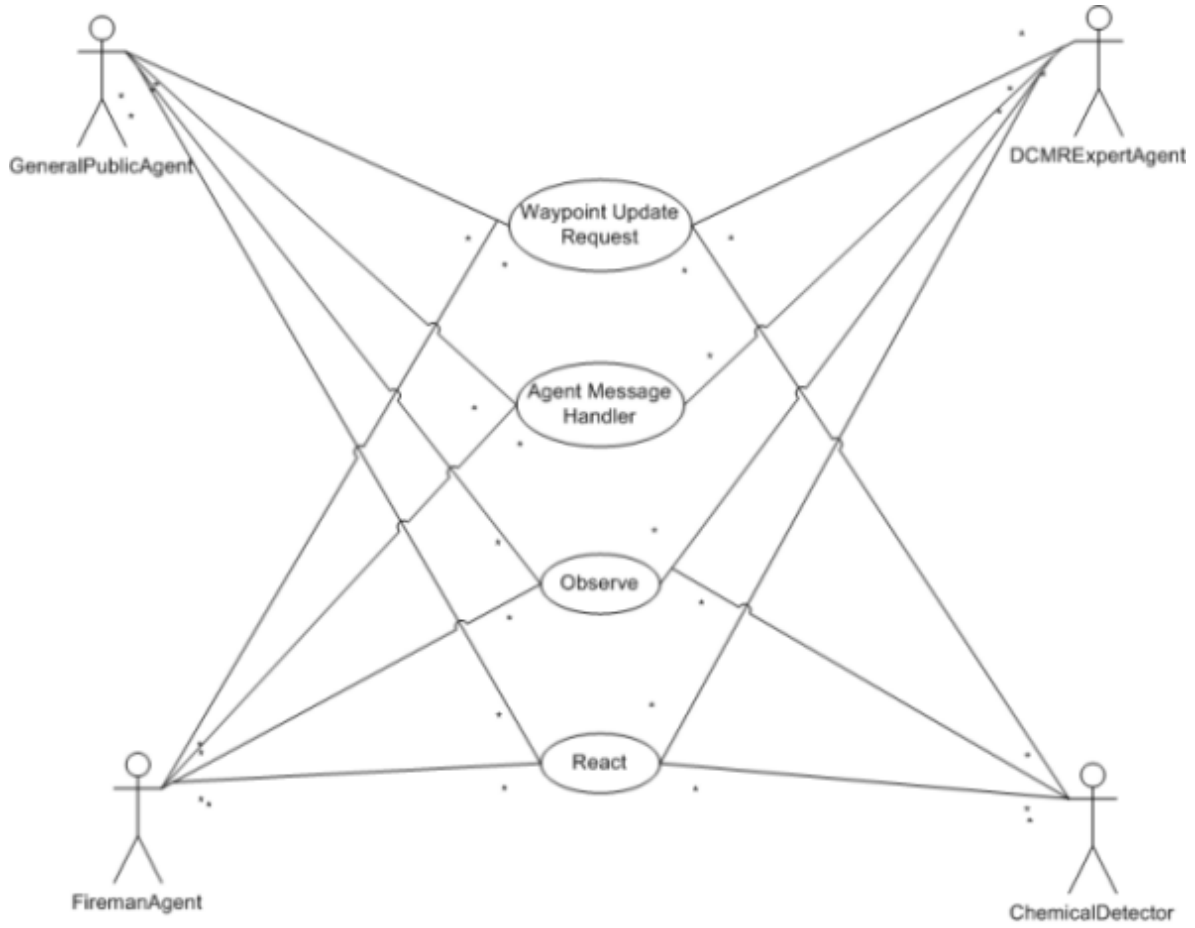


Figure 4.4: Use Cases Participating Agents

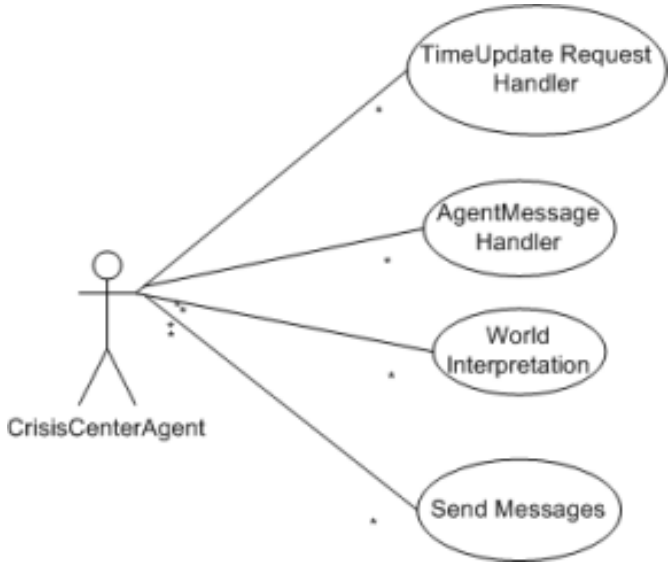


Figure 4.5: Use Cases Crisis Center

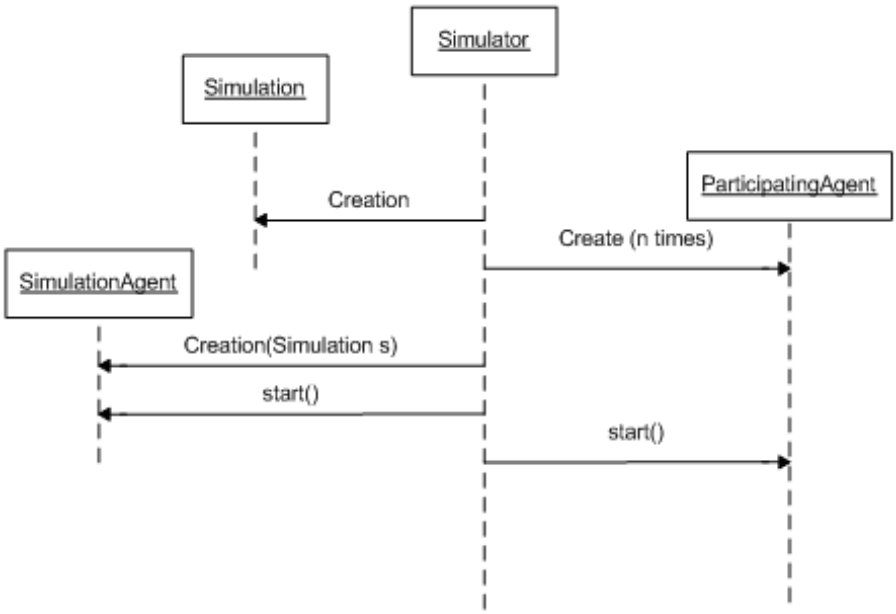


Figure 4.6: Creation of the Simulation

4.2.2 World Updating Components

Before we explain the updating algorithms in detail, we first explain the basic building blocks of the simulation system, and the general ideas behind the updating schemes. Then we will discuss the general implementation of the used updating schemes.

Simulator

The main point of the application focuses around the Simulator class. The simulator starts all parts of the system, the world, the agents and the agent communication and expert system intelligence. In the Simulation we can also indicate how many agents of different available kinds we need. The basic concept of the simulation is that during a fixed time interval calculations take place to update all the Waypoints and actions of the agents in the world. When all those actions are processed, new information is "shared" between the different agents and the crisis center. The crisis center will then give a new hypothesis to the user. For now we focus on updating the world.

For making the updating scheme work and to be not too computationally heavy, we need do develop a scheme that can process data fast, and that does not do too much unnecessary updating, for instance replacing a 0 with a 0 during the entire simulation is not the most useful update. In some cases these kind of updates can be avoided through several schemes. To explain the world update scheme all parts of it will be addressed. First the components that need to be updated will be addressed. While discussing the actual update algorithms the components that are responsible for the actual updating and their function will be discussed.

First we explain the world in a little bit more detail. In the world, we have knowledge of the dimensions of the world, and therefore also about the number of waypoints that are operational. Waypoints will be placed in a fixed distance from each other and they will count for the area directly surrounding that point. At this point not too many details are modeled in the world because it would make the physical model more complicated. But if you want to make agents observe certain things, it is always possible to

add new objects to observe. Mostly though, the Waypoints will contain numerical values that represent certain environment variables.

World + Waypoints

To make the world as easy to process as possible during data creation and still make it possible to import the features of the world that need to be in there for the simulation, certain compromises have to be made. The world is being made quite simple, but it is designed in such a way that it can be expanded to the level of complexity that is required for any given simulation.

The World consists of a fixed number of waypoints that is based on two predefined constants: the world size and the map grid. These Waypoints are stored in a two-dimensional array. The world size is defined in meters, as is the map grid. The coordinate for now is a Cartesian (x, y)-based map, with the origin in the lower left corner. The borders of the world are stored in the values MapsizeX and MapsizeY.

In the future this could be changed into GPS coordinates if necessary, but as the world as yet is not representing any specific location in the real world, for now meters are used as the measurement unit in the world. All waypoints are located a fixed number of meters from each other. The distance between those waypoints is determined by the constant MAPGRID (see Figure 4.7).

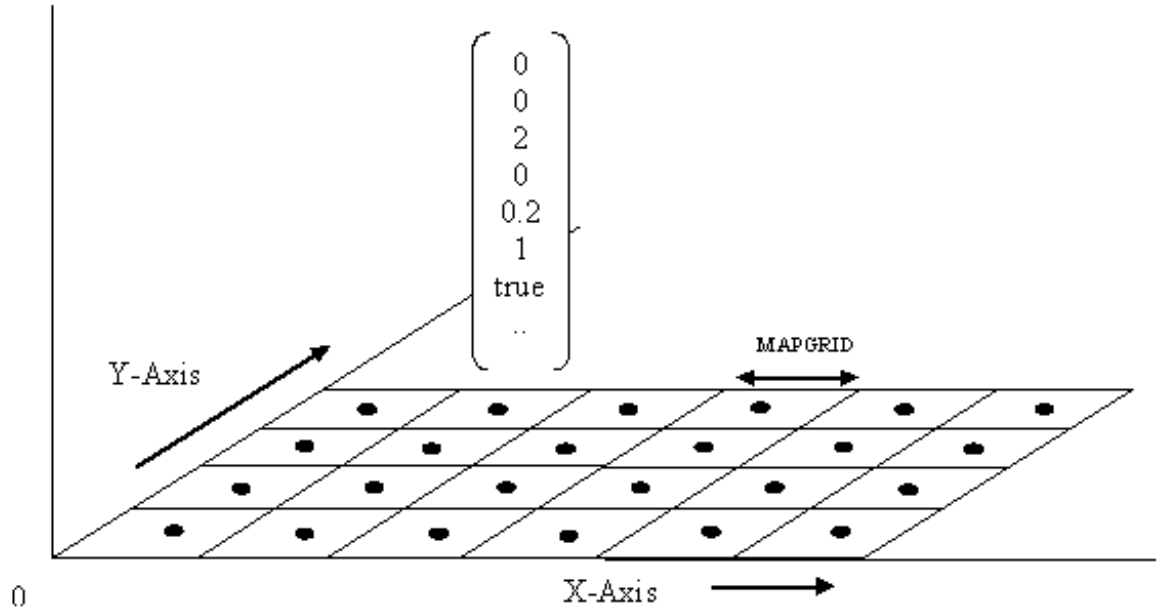


Figure 4.7: Simulation World Design

Every Waypoint in the world consists of a finite number of data. This data is represented in ints, doubles, and booleans, and whatever is the most convenient data type to use, is used. This is of course dependent on the number of states that are needed to represent the necessary values. Each Waypoint's values count as the value for the square with size Mapgrid around it.

Table 4.1 gives alist of values that are currently available in a waypoint (The number and types of the data are subject to constant change and have to be updated here as well).

Table 4.1: Waypoint Values and their Meaning

Type + Name	**Meaning**
double locationX, locationY	Where is the waypoint located. X and Y are represented in meters.
double temperature	The current outdoor temperature in degrees Celsius
double windSpeed	The current windspeed in m/s. In this simulation we have one default windspeed for the entire world, but if a more complex wind speed model needs to be implemented, it is possible.
int visibility	0: visibility <= 5 m 1: < 10 m 2: < 100 m 3: < 1000 m 4: > 1000 m
boolean bang	Hear a bang or not
boolean flashLight	See a flashlight or not
Vector gasVector	For each gas available as a simulation a set of properties are being set up (see Table 4.2).
int specialObjects	This value can represent special objects that are visible in the world for the agents, not directly related to a certain gas that disperses. Future examples could be water (a river or a lake or a flooding could be represented as such), high buildings, trees and solid or liquid chemicals that are on the floor near the scene of an incident. Currently this value is not yet in use.
double fireLevel	The firelevel is a value between 0 an 1 and represents the amount of fire that is going on on a certain location. 0 means there is no fire at all, 1 is a very heavy fire. (more on this in the paragraph on fire modeling)
double smokeLevel	See fireLevel, only this time with level of smoke
int windDirection	Wind Direction is an int with range 0 to 15 that represents the wind direction on a certain waypoint. The directions are appointed clockwise starting with 0 = north, 1 = north-northeast etc.

Table 4.2 describes what kinds of values are being stored inside a GasVector, which is basically a vector of Gas objects. Each Gas object has its own set of values. These are the following:

Table 4.2: Gas Object Values and their Meaning

Type + Name	**Meaning**
double gasConcentration	Gas concentration in units PPM
int gasColor	The color of a certain gas that is spreading. The Colors are defined as Ontology data (more about this later in the chapter).
int gasBehaviour	In this value, special unique properties of a gas can be represented. Currently this value is not used, but is reserved for future use.
boolean gasVisibility	States whether the gas is visible or not.
int gasSmell	Different types of smell can be defined for a certain gas. Currently used examples are: BITTER_ALMONDS, SHARP_ODOR and ROTTEN_EGGS.
int gasType	The gases currently implemented in the system: 0: Cyanide 1: HCl 2: Methylmercaptan 3: Nitrogentetroxide 4: Trichlorosilane 5: Toluene 6: Ethylene 7: Butylmercaptan 8: Sulfurdioxide 9: Ozone
boolean gasDanger	Indicates whether a gas is dangerous. This is the general dangerous boolean flag, that gives room to elaborate on danger details, listed below.
boolean gasDangerPoison	Determines wether a gas is poisonous.
boolean gasDangerBurningEyes	Determines wether a gas causes burning eyes.
boolean gasDangerInhalationProblems	Determines wether a gas gives inhalation problems.(this value can only be set when gasDanger is set)
boolean gasDangerFlammable	Determines wether a gas is flammable.
String gasName	The name of the gas
double MolecularMass	Needed for GPM calculation

When the simulator is created, it receives all information to get the simulation started, e.g. the number of firemen, the size of the general public in the scenario, the starting time of the scenario and the end time of the scenario (see Figure 4.8). Then the World and ScriptControl class is created, which handles the unfolding of the script. When this is done, the agents that are walking around in the world are placed in the world. This means that they are given the appropriate coordinates to get a legitimate place in the world. In the picture this is shown schematically, because in reality the simulator will create JADE

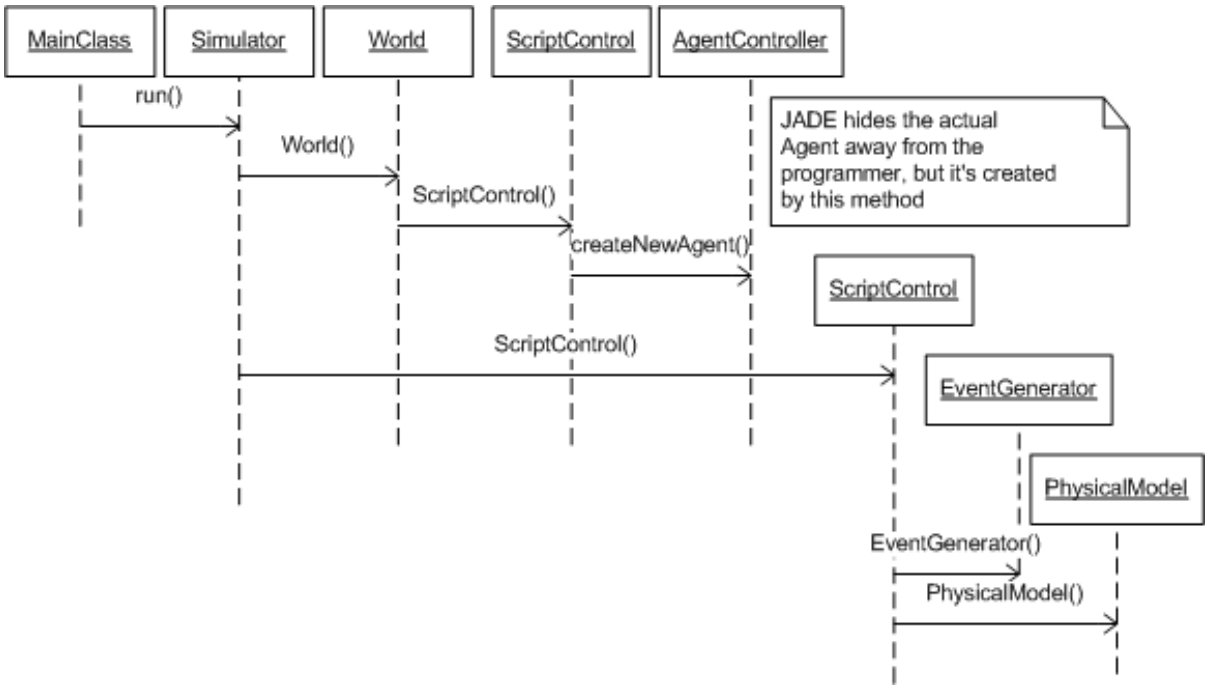


Figure 4.8: Scenario and Agent Creation

objects called AgentControllers for each agent to be created. These agent controllers are then placed inside the Agent Management System [Cai03] and the Agent platform is being started.

Because the Simulator is a subclass of Thread, when the start() method is called, the run() method is being executed. This run method controls the updating scheme of the world. It updates the World according to a script (controlled by the script controller) from the start time of the simulation to the end time of the simulation. For the time in the simulation a Special Minute-based Time class has been created, called SimulationTime. It contains minutes, hours and daysSinceStart, which means that it keeps track of the times that the time has passed 0:00, i.e., new days have started.

Simulation Constants

One of the nice features of Java 1.5 (the programming environment used) is the possibility of making a class in which globally used values/constants can be stored, which can be statically imported in to the class that wants to use it. This is quite practical if the system makes use of different agents that all have to use the same constants to fill in e.g. messages with standardized content. That is why we have created a class called SimulationConstants that contains all the globally used constants. This makes it very easy to use constants in parts of the code as it makes the code very readable. Instead of using the class name and package, you can now just use the name of the constant.

4.2.3 World Updating Algorithm

In the Simulator it is possible to generate different scenarios that can take place in your world. This raises the question how to implement the scenarios in the system. There must be a place where a scenario that has been requested by the user is being stored and executed. In the case of MACSIM this is a script. In our system a script is a chain of events that is monitored according to the duration of the simulation. It looks a bit like a timeline. The general scenario (i.e. dynamic script unfolding) takes place inside something called the Script Control. During unfolding of a script, several different events can take place in the course of time. Those different events are represented by different event generators

that can calculate the effects on the values of the Waypoints in the world caused by that particular event. An event generator calls the necessary formulae from the physical model to receive the most recent value and replace it with the earlier one.

The second reason to use different event generators is a modeling issue. Different events ask for different models and also for different update schemes. Gas dispersion for instance (it also holds for other fast-moving events like the sound or light flash accompanying an explosion, only in these cases it's almost an instantaneous event and therefore not a striking example) is implemented using a Gaussian Plume Model, that has to update all the Waypoints during every time step. Because gas is very light, depending on the wind speed it moves very fast across the world. The chance that a certain Waypoint is affected during the next time step is therefore always there, even when it's far away. This update mechanism takes a lot of time but with a model describing a phenomenon like this, it's unavoidable. One cannot take the risk of skipping Waypoints because it won't be affected anyhow.

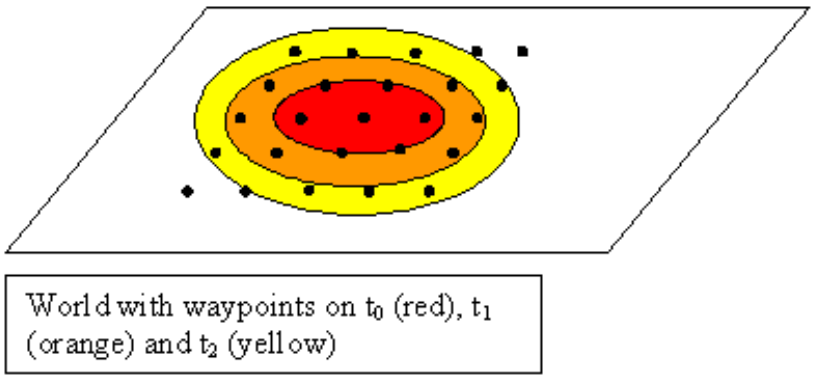


Figure 4.9: Explanation Dispersion Concept

This last statement does not hold for, for instance, fires. Fire can spread fast, however it cannot float through the air, like a gas cloud. Therefore when a fire starts at waypoint (x, y), during the next time step, only a couple of Waypoints surrounding the fire will be at risk and we can safely ignore the Waypoints that are in an entirely other area of the world. Of course, Waypoints downwind of point (x, y) have more risk of catching fire than others, but places on the other side of the map are of no interest to us.

In this kind of event modeling it is safe to ignore those unlikely Waypoints and only update those that are of interest. This principle works as follows. First a fire takes place at point (x, y). In the next time step, you look at the Waypoints directly surrounding (x, y) and they are put in a list. Then those Waypoints are updated and you save the list for the next time step. During each time step the list grows. Every time step new Waypoints are being added for updating. In this way many Waypoints need to be updated, but certainly not all of them.
When all the effects of all the events on the world are calculated, it's time to release the updated world so the agents can observe it. The scenario controller combines all results of the calculations and releases the updated world. The updated world is then given back to the Simulator, where it's now in the hands of the agents to observe and react to it. Of course they still have to receive the waypoint values, but that is being discussed in the section about the SimulationAgent.

The class that controls the entire simulation (as well as the main program) is the Simulator class. This is the class that is being started in the Main class when the application is started. As the Simulator class

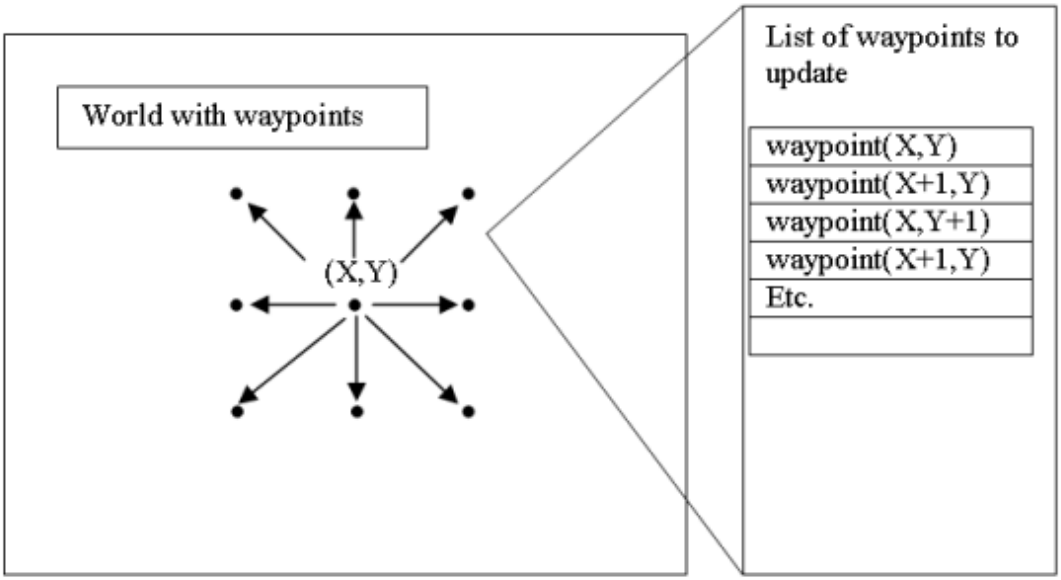


Figure 4.10: Explanation Expansion Concept

is a Thread, the simulator is a separate process that is independent form the Mainclass. The Mainclass therefore is reduced to a sort of static bootstrap class for starting the system, while the actual application code is placed in the Simulator class.

The Simulator class keeps track of the main simulationTime. For some features though, the time since a certain event happened or the time certain events take to unfold is needed. Therefore there also exists a RelativeTime class, which contains days, hours and seconds. It is possible to compare a RelativeTime object to another RelativeTime object, as well as comparing a SimulationTime to another Simulation-Time. When the end of the simulation is reached, the updating process stops, and the Simulator thread is stopped.

Controlling Scripts

The Script Control works as follows. It "reads" a script. In this script certain events are noted. An example of a script could be something like the strucutre given in Figure 4.12.
The script controller reads this script, and whenever it's the right time (e.g. 14:11) it "launches" two Event generators. One for the Loud_bang, the other one for the Gas_dispersion. Those active event generators will be stored inside the ScriptControl and during every update run, it will ask for updated values from all active Event generators. Some events are fairly short (for instance the Loud_Bang scenario), and to avoid too much unnecessary calculation we first ask the event generator responsible for a certain event if it's still doing anything. The event generator knows this because he knows it's starting time and the current time, so for instance, when it's 14:40, a loud bang that started at 14:11 will not be active so the values of that event generator will not be bringing anymore new values, therefore we skip that particular event generator, and we continue to an event generator that is still active.

So Waypoints in the world are updated by event generators. In an EventGenerator events are generated and those events manifest themselves to the agents in the world as changes in the Waypoints' values. The event generator does not know anything about how to calculate the values of particular Waypoints, he leaves that to the Physical Model, where all the formulae that need to be used are stored. The event

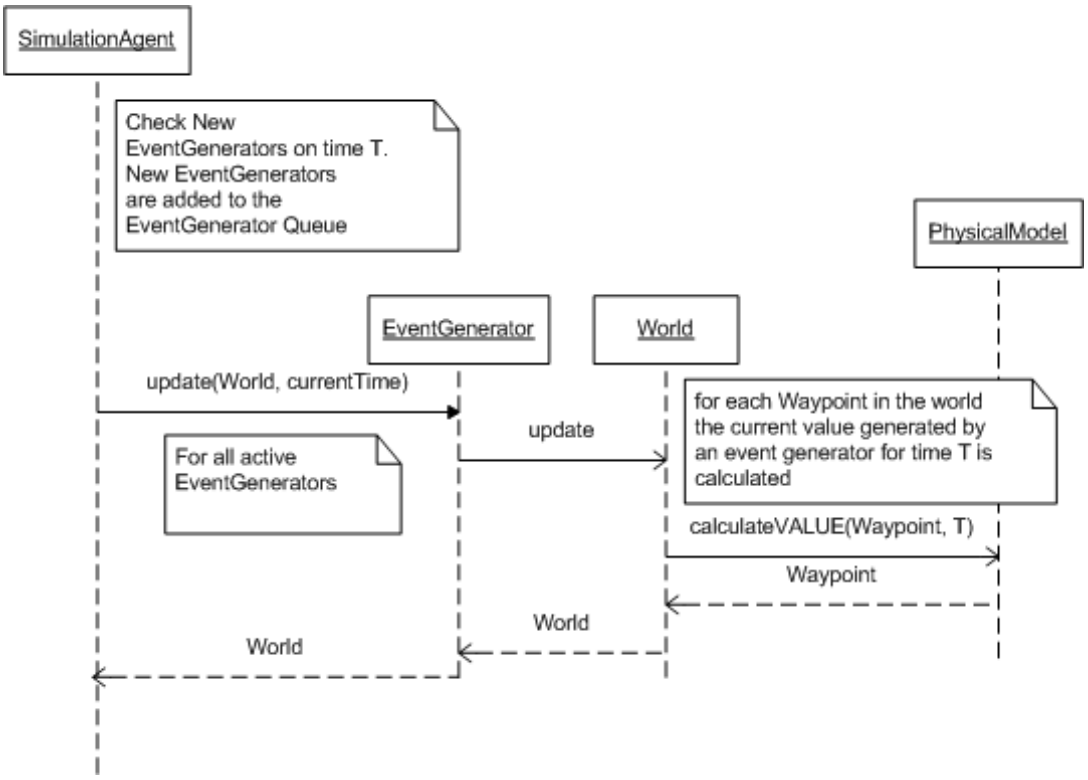


Figure 4.11: World Update Sequence Diagram

```
Scenario Start: 1400
Flash 1410 loc(x,y)
Loud_Bang 1411 loc(x,y)
Gas_dispersion 1411 loc(x,y)
Small_fire 1445 loc(u,v)
Etc…
```

Figure 4.12: Example of a Script

generators can generate all kinds of events, from gas dispersion to fires or loud noises or flashes of light. Everything is possible as long as it can be represented in numerical values that can be stored inside the Waypoints. This makes it very easy to update.

The reason to use several event generators is twofold. Firstly with this mechanism one can develop several events at the same time. It is very important to stress at this point that stating that several events can take place at the same time does not mean that those events are influencing each other too much in the world model. In the real world, it is common that whenever several disasters happen at the same time, this leads to special situation that in a computer model is very difficult to cope with. Consider for instance the situation that a fire breaks out and at the same time gas disperses. This dispersing gas appears to be highly flammable, so when the dispersion cloud reaches the fire, this leads to an uncontrolled explosion that is very difficult to model, especially because the gas dispersion model is also not valid anymore after this explosion (the dispersing gas is gone). In the system one must combine the effects of different events, but in such a way that they don't influence each other too much. We can take care of the combination of heat from different fires or smoke coming from two sources by using

some sort of averaging function, but things like explosions that are the result of two separate events interfering with each other will not be possible.

The update routine that is executed every minute during the simulation time interval works as follows:

- The Script controller is notified that a new time instance has started. Based on this information the scriptcontroller executes Eventgenerators that might need to be activated at this point.
- The world receives new values from all the event generators through the ScriptControl.
- The agents receive the new values for the waypoint they are closest to.
- The agents respond to the situation created in the new time instance.

The Scriptcontrol class makes sure that the script is being executed. A script is defined by a series of events, represented as Event class files. An Event has a SimulationTime, a description that describes the type of event, and an x and y coordinate where the event's starting point is.

When the current time in the simulation routine matches with the starting point of a new event in the script (Figure 4.12), the script control "launches" a new event and does this by adding a new eventGenerator with the new Event as a parameter to a set of active EventGenerators. These active EventGenerators are the objects that enable the world to be updated. During the update cycle, each active EventGenerator is being asked if it's still active. This is checked through checking if the standard duration time of a certain event is equal to the RelativeTime from the starting time of the event to the current time. If it isn't active anymore, it's removed from the set of active EventGenerator objects. If it's still active each Event generator is being asked to update the event that they generate, that is to update the Waypoints in the world with new values, according to the new time step. This is done with help from the Physical Model, and with help from two different updating mechanisms, Dispersion and Expansion.

Each Waypoint is updated individually, but the amount of Waypoints that need to be updated by the EventGenerator can vary per scenario and of course it depends on the updating scheme of that particular event. This is the division of used schemes so far:

Table 4.3: Currently Available Event Types

Event Type	**Update Scheme**
Loud Bang	Dispersion
Flash	Dispersion
Gas Escape	Dispersion
Light Fire	Expansion
Heavy Fire	Expansion

When based on the event type an updating scheme has been chosen, the EventGenerator starts to update the Waypoints in the world. The EventGenerator does this with help from the PhysicalModel class, which contains all formulae that the simulation needs to update the Waypoints. This class therefore contains the necessary methods to update every Waypoint value that is necessary.

Dispersion Updating

The dispersion model is updated as follows:

- first the relative time t of the incident is calculated, based on the current time and the starting time of the incident.
- then t is used in the formula for a specific event for each individual Waypoint.

- formula in the PhysicalModel produces an updated Waypoint (with updated values for that specific event).
- the updated waypoint is returned to the world.

Dispersion currently is used for generating a flash and a bang, which spreads simultaneously a flash and a bang to all waypoints for one time instance. It basically switches the sound or light boolean on or off in a Waypoint. This is not a very advanced example of the dispersion algorithm. Therefore we explain the dispersion through gas dispersion. As has been said in the design chapter, for gas Dispersion the Gaussian Plume Model or GPM is used.

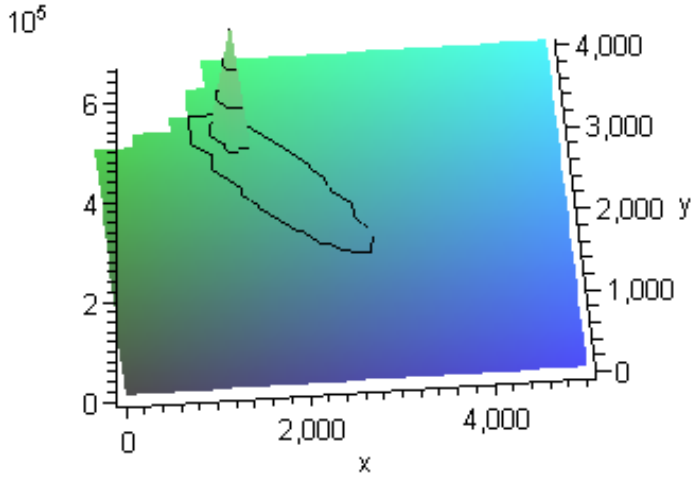


Figure 4.13: 3D Plot of GPM Model [CPR92]

In Figure 4.13 we can see how the concentrations of a gas cloud are being spread across an area, on a height of 2 meters. This is approximately the height on which the agents that are walking around in the world will be experiencing the gas concentration levels, given that the dispersion source is on ground level. To determine the weather conditions this model uses the Pasquill Stability classes as used in the ALOHA model. The values vary from A (very unstable) to F (very stable). For the Netherlands the Royal Dutch Meteorological Institute (KNMI) made a reference table for what Stability Class one should use during a certain time of day/year and the wind speed [CPR92] (see Figure 4.14).

The actual model comes down to a calculation of a concentration on a location (x, y, z) in which x,y and z are relative coordinates from the dispersion source in the sense that the x-axis is the wind direction with the origin in the dispersion source point and the cross wind (side wind) direction is the y-axis, also with the origin at the dispersion source. The concentration on a time t for $(t > 0)$ is:

$$c(x,y,z,t) = \frac{m}{(2\pi)^{\frac{3}{2}}\sigma_x\sigma_y\sigma_z} \cdot \exp\left(-\frac{(x-u_w t)^2}{2(\sigma_x)^2}\right) \cdot \exp\left(-\frac{y^2}{2(\sigma_y)^2}\right) \cdot \left(\exp\left(-\frac{(z-h)^2}{2(\sigma_z)^2}\right) + \exp\left(-\frac{(z+h)^2}{2(\sigma_z)^2}\right)\right)$$

In this equation, m is the released mass, u_w is the wind speed and h is the source height. σ_x, σ_y and σ_z, the dispersion parameters, depend on the wind speed uw and the time of year, as they depend on

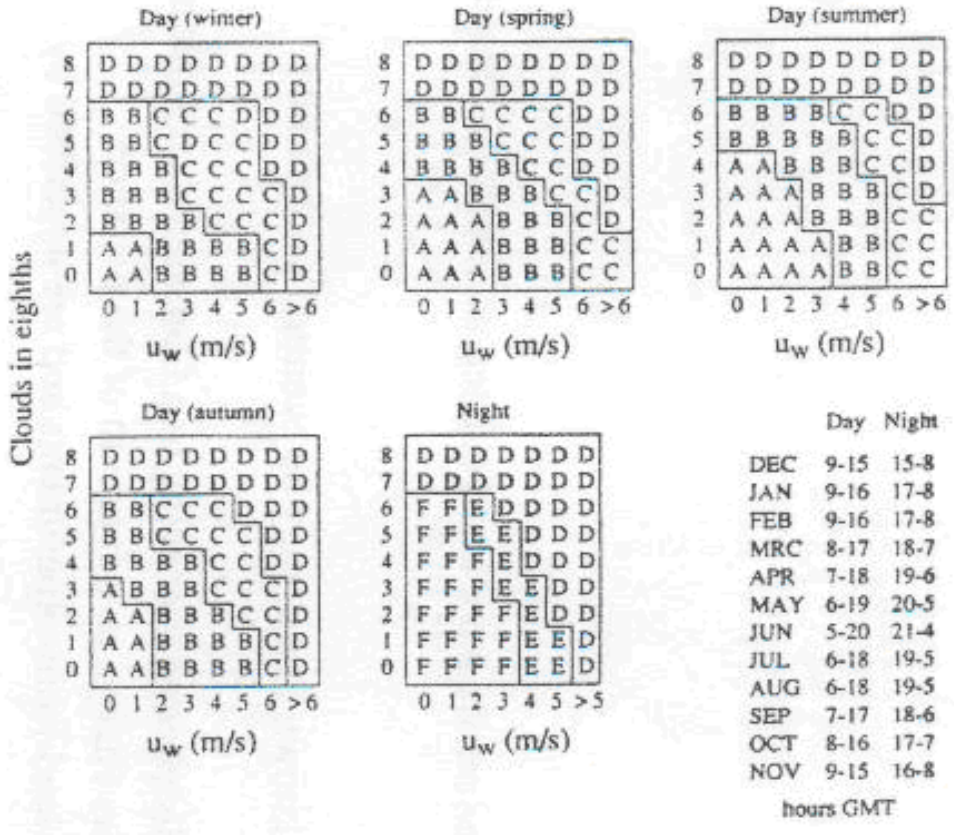


Figure 4.14: Pasquill Classes A to F for The Netherlands [CPR92]

constants a to f that are dependant on the Pasquill stability class. They are calculated as follows:

$$\sigma_x = ex^f$$

$$\sigma_y = ax^b$$

$$\sigma_z = cx^d$$

The values a to f are obtained by getting the right values out of Table 4.4:

Table 4.4: Dispersion Parameters

Stability Class	a	b	c	d	e	f
Class A	0.2635	0.865	0.28	0.9	0.13	1
Class B	0.1855	0.866	0.23	0.85	0.13	1
Class C	0.1045	0.897	0.22	0.8	0.13	1
Class D	0.064	0.905	0.2	0.76	0.13	1
Class E	0.049	0.902	0.15	0.73	0.13	1
Class F	0.0325	0.902	0.12	0.67	0.13	1

The sigma constants are based on climatological parameters, such as air stability and wind speed. x and y coordinates are translated and rotated to x_r and y_r coordinates, in which the x_r is the downwind axis and y_r the crosswind distance. This translation is based on the wind direction in the world, set in degrees (Figure 4.15). The translation uses the following formula to convert the wind angle β in degrees

to a rotation angle α in radians:

$$\alpha = \frac{(\beta + 90) \mod (360)}{360} \cdot (2\pi)$$

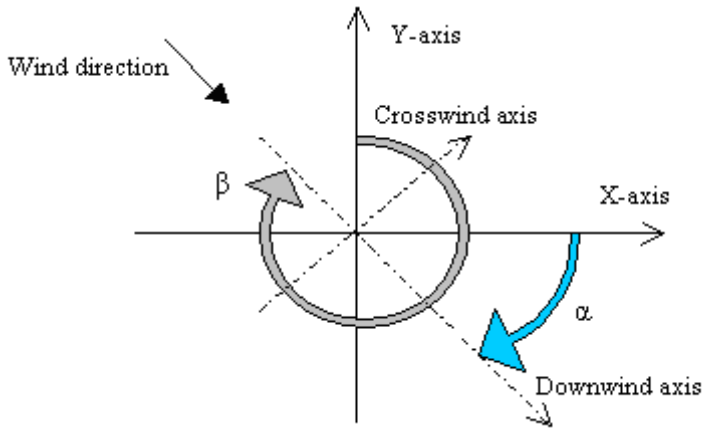


Figure 4.15: Conversion from Wind Direction to Rotation Angle

With the correct rotation angle α, these formulae are used for coordinate rotations to obtain a x_r and y_r in the formula given an x and y and the origin coordinates of the event x_o and y_o:

$$x_r = (x - x_o) * \cos(-\alpha) + (y - y_o) * \sin(-\alpha)$$

$$y_r = -(x - x_o) * \sin(-\alpha) + (y - y_o) * \cos(-\alpha)$$

Through this formulae we now know the x_r and y_r coordinates to determine the concentration in kg/m^3. They are being evaluated in the GPM-formula as the new x and y, with z 2 meter off the ground, because that is the height from where the dispersed gas will affect the general public mostly. The only thing still left to do is to convert the concentration from kg/m^3 to units PPM, which is more regularly used in gas concentration measurements. This is done through the following formula (which holds for standard chemical circumstances) [dJ68]:

$$c_{ppm} = c_{kg/m^3} * (10^6) * \frac{24}{M_{gas}}$$

M_{gas} in this formula means the molecular mass of the gas for which the concentration is being calculated, 24 is used because one mole of gas represents 24 l in standard chemical conditions and the 10^6 multiplier is used to make it count the number of gas particles per milloin partices. When the correct concentration has been retrieved, the value of the concentration is updated and if necessary several other gas parameters in the waypoint are updated for every Waypoint in the world. Whenever the value is so small that it cannot be registered in reality in the first place, It will be kept 0. Every waypoint in the world is run through to calculate this value, so you sometimes can run the risk of updating a zero with a zero, but then again it is a very accurate formula and it's simpler to just do this obsolete updating than to do a computationally complicated prediction to predict in advance which values are going to be 0 based on the windspeed and relative time.

Expansion Updating

For events that do not spread as fast as gas particles or light or sound, it is easy to predict which waypoints need to be updated. For this kind of spreading we have devised the expansion algorithm.

Expansion has a certain depth d, which means that Waypoints with a distance d Waypoints away from the source of the event are affected. Only they will not all be affected at the same time. For the expansion to fully develop expansion times exist, that enable incidents to spread over a certain time. After the 0-expansion time the incident spreads to the adjacent Waypoints with a depth 1. This could go on until depth d, depending on the type of incident (See Figure 4.16).

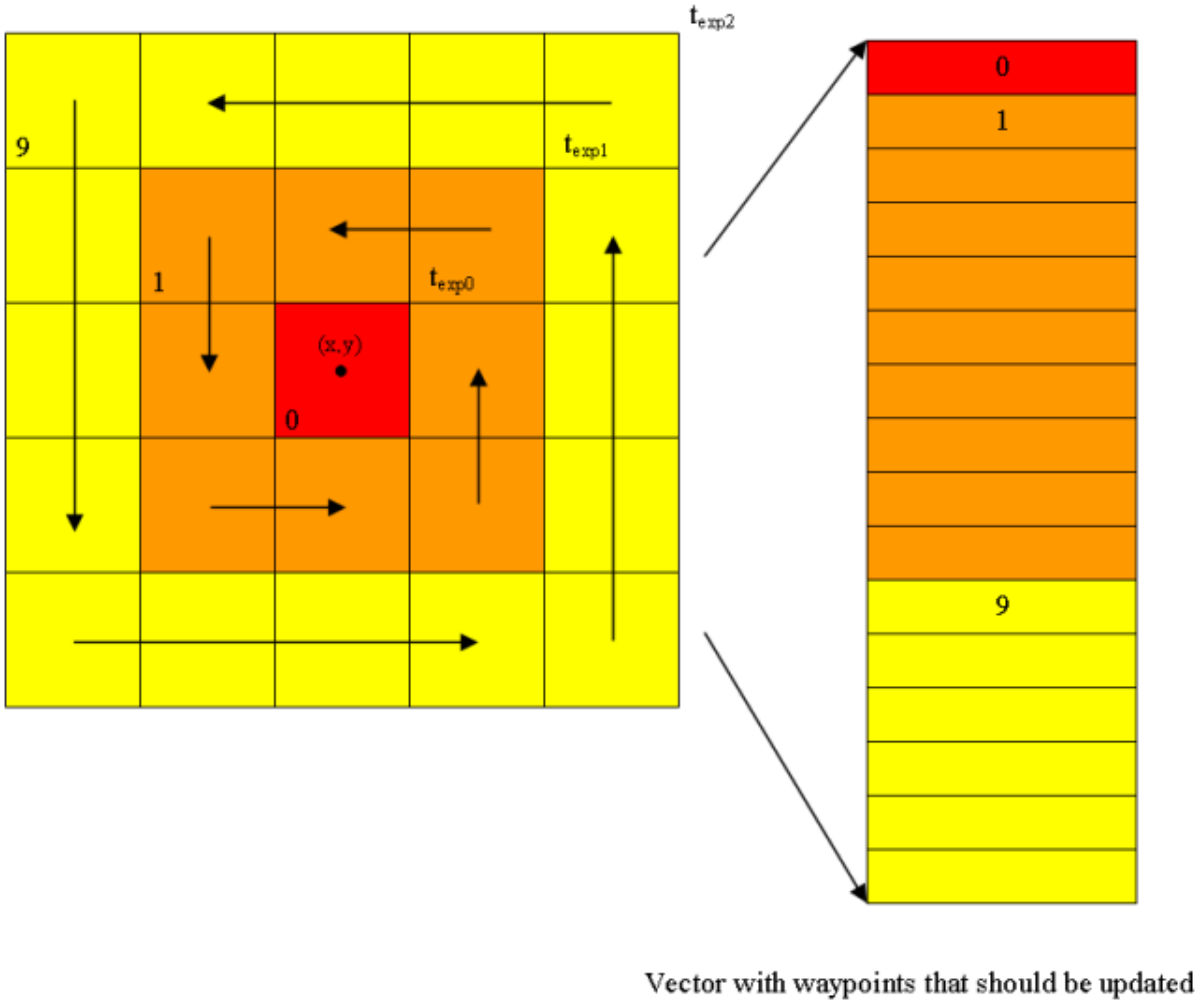


Figure 4.16: Explanation Expansion Algorithm

From a source Waypoint (x, y) the values are being updated. After an event specific RelativeTime t_{exp0}, the event expands to the 8 waypoints surrounding the source. The Waypoints that need to be updated are added to a list of Waypoints that need to be updated, including the source. Those are the only Waypoints that are updated. Each new series of Waypoints that is added is added as follows: first the top right Waypoint adjacent to the source, and then the other ones following in a counter clockwise fashion.

Later, when the RelativeTime t_{exp1} has passed, the same scheme is used to add a new set of Waypoints to the waypoints ready to be updated; again it starts with the top left adjacent Waypoint, and then the next one counterclockwise. Only the Waypoints that are in the list are updated, so this should considerably reduce updating times for such Events, because of reduction of waypoints that are not in the list.

4.3 Multi-Agent System Design

When we are going to discuss the Multi-Agent component of MACSIM, it is unavoidable to discuss some fundamentals of JADE as well, because the Multi-Agent component of MACSIM was built upon JADE. That's why we first will discuss the JADE architecture that was necessary for the agents to run. This will not be a comprehensive manual on how to program JADE-based multi-agent software. For this very good documentation is available from [CC04], [BCTR05] and [BPR01]. We will however discuss those sections of JADE that are required to explain our design of the data flow in the system and the behaviors as they are implemented currently.

After that the use cases of each agent will be discussed. Sometimes these usecases will result in the same implementation for all agents, sometimes the implementation will differ on individual details. Depending on the general applicability of a use case or Behaviour for the agents, the Behaviours will be treated at once or per agent individually.

4.3.1 JADE Framework

In this section the design of the Crisis simulator is described with the emphasis on the part of it that was implemented in JADE. JADE of course stands for Java Agent Development Environment and has more or less become the standard for independent Agent software, with asynchronous Messaging. The JADE version that is used in the Crisis Simulator is JADE 3.4. JADE enables programmers to create an environment in which agents can be added and removed in real-time. Because the simulation that we use is being created in advance, the JADE platform only comes in to play in runtime. JADE version 3.4 has an in-process interface that helps programmers to start up a JADE platform instance while running user-generated code.

Therefore we can create a Specific simulation that fits our requirements before JADE, and the actual simulation is being run. We can fill in any properties we want into an interface that gives a description of the simulation that is about to be run. We put all the necessary data into a Simulation object that, via the in-process interface of JADE, can then be incorporated with an Agent during startup.

Originally, when the simulation part of the code was designed, it was vital to check whether the simulation and the expansion algorithms were functioning and if the agents were doing what they supposed to be doing. Therefore first a design was made in which the simulator class had a sort of directing role and in which JADE functionality was not yet incorporated. The Simulator ran the Simulation, it created the "agents" (because they were not implemented as JADE agents you might not speak about them in terms of agents now). The Simulator was conducting even all data transport. The design had to be changed however, because JADE does not allow programmers direct contact with instances of JADE Agents, because of the demand of independency and autonomous agents. It only enables the programmer to contact Agents through AgentControllers, which allow some form of communication, but not enough to maintain a design in which the Waypoints are given to an agent after an update cycle. Therefore the simulator could not directly provide the agents with freshly updated Waypoints each time the Waypoints update routine was called. Therefore another construction had to be devised and the solution was found in a SimulationAgent, which was a JADE Agent that formed the bridge between the Simulation that was updating constantly, and the JADE Agents that needed Waypoints in order to interpret the environment and respond to it.

It was then decided to call the agents that were participating in the simulation ParticipatingAgents, and let all the agents that were around be extensions of the abstract ParticipatingAgent class, that received updates of the environment from the SimulationAgent, which is a different type of Agent because he is not Participating in the simulation, it providing a gateway to the simulation itself. If the simulationAgent

was running the simulation, then the Simulator object became less dominant. Its main function became the starting of the JADE platform and creating of the agents and running the Interface that provided the simulationAgent with the data for the simulator. In a way it became more a supporting class.

At this point in MACSIM development, the simulation is a proof of concept, in which the system will

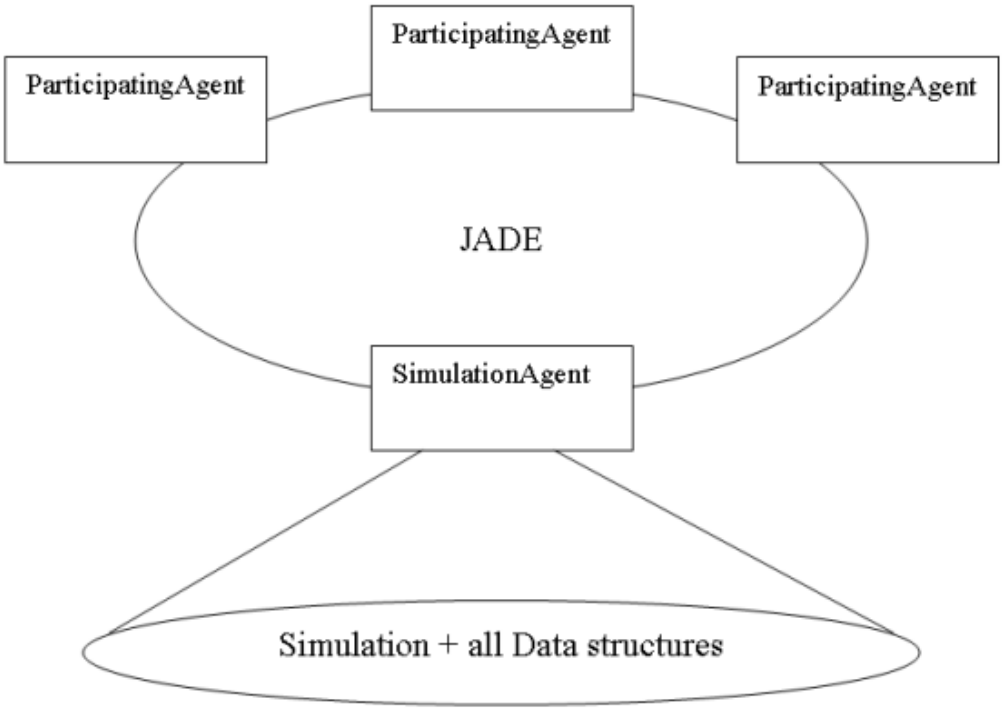


Figure 4.17: JADE SimulationAgent Architecture

be running on a single PC. Later it should become possible, in compliance with the global design and proposed solution in section 1.3 to add for instance mobile agents to the system. For more information about the current system configuration please refer to the chapter on Implementation.

4.3.2 JADE Message Protocol

If the agents want to communicate via the JADE Framework, they should have a protocol at their disposal that enables them to exchange status information at their current location. The protocol currently implemented in the crisis simulator takes care of the basic requirements for exchanging messages between the ParticipatingAgents. This protocol is *not* used for exchanging Waypoint information with the SimulationAgent, as the SimulationAgent does not play a participant role in the crisis simulation. For now this protocol is used by the CrisisCenterAgent, the ChemicalDetectorAgent, the FiremanAgent and the DCMRExpertAgent.

Requirements

There were a couple of requirements for the protocol. The first one was that it should be able to support Siska Fitrianie's World Ontology as message content [FR06]). Her world model consists of a lot of concepts and situations that can be going on inside a simulation world, so using her ontology would create a semantically rich environment in which, in later versions of the crisis simulator, a large range of events can be simulated, and therefore also a large range of different messages types can be created.

Because her World Model was made inside Protégé, the BeanGenerator can be used to generate the required ontology relations for JADE to support World Model messages.

Secondly, we have to make sure that the message traffic takes place as efficiently as possible. In real life, the communication in case of a disaster is purposely kept very efficient because every second counts in crisis situations. It is not recommended to waste time with obsolete messages that are already being sent. Therefore the messaging should be minimized.

The first priority for the system should be for the agents in the field to send messages to the crisis center. They have to report about what they see or hear and that can be considered as a situation. At first it's only about reporting. Later on requests, queries and orders can be transferred between the agents themselves. For now, the main focus is on the informing of the crisis center and the crisis center giving feedback about these messages. This does not mean, however, that the message protocol implemented should not support inter-agent communication. For future use, it's required that the agents should be able to maintain inter-agent communication if it's required for their agent behavior.

Constraints

Some of the information inside Fitrianie's World model is still too elaborate to use in a very simple crisis simulation as it is now. So for now some parameters in the messages will be given a default stub value, but during redesigning of the Simulation component, it can be designed in such a way that it is able to use the full functionality and semantic richness that is being given by this World Model.

For now, the functionality of the message protocol is limited to the sending of one message per time instance (in this case minutes). In this way the decision process is being simplified, because it saves a lot of overhead filtering duplicate messages. Of course when the system starts to become a system that runs on different computers (in the first setup it runs on a single PC), this constraint should be removed but for now it is a legitimate constraint, because running all agents on the same agent platform guarantees that a message that is being sent is also being received so no acknowledges or confirmation messages are required.

Another constraint in the protocol is the number of messages that can be processed in a time instance (in this case a time instance is a minute.) This is different per agent, because in real life, for instance, the crisis center is capable of processing many more messages than a fireman in action. Therefore it could also happen that messages will not reach the agent concerned because he doesn't have time to process all messages that are sent to him. To be more precise, an agent does receive all messages, but it will not always have the opportunity to process all messages. This will be depending of the agent's ability to focus on the messageQueue.

4.4 Agent Behaviors

In the class diagram in Figure 4.18, we can see that there are two different types of agents. The SimulationAgent is a direct extension of a JADE abstract Agent, while the other agents are implementations of the class ParticipatingAgent. This seems to be a logical taxonomy because of the fact the ParticipatingAgents are participating in the simulation and the simulation is only a data provider. ParticipatingAgents, however have properties like coordinates and a reasoning engine, but from a certain perspective, the CrisisCenter to an extent is the odd one out. It doesn't have coordinates and doesn't have any direct sensory input. It gets its information from the other participating agents in the field. It therefore does not use several features of a ParticiaptingAgent, but still is considered to be one. Maybe later on the class

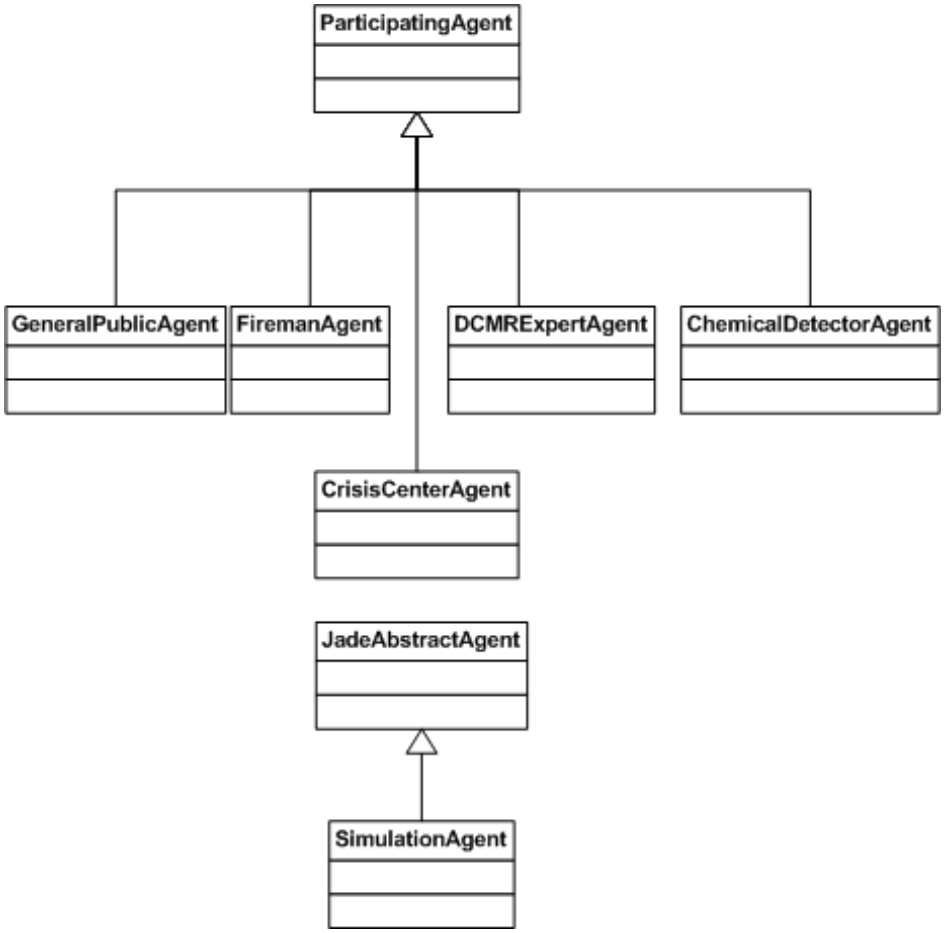


Figure 4.18: Class Diagram MACSIM Agents

taxonomy could be changed in such a way that agents in the field are a special subclass of Participating agents so the Crisis Center will not have obsolete features. This is also the reason that the Behaviours of the ParticipatingAgents are treated in another section than the Behaviours of the CrisisCenter.

JADE has a lot of possibilities to include Agent Behaviours flexibly in the EventQueue of a system. Because of the fact that the system needs to be synchronized for updating and because the Participating agents need to execute actions that are supposed to be happening on the same time simultaneously, we deliberately chose for an architecture in which the agents follow a set of steps in which each step is represented by a Behaviour. This Behaviour setup is being supported by JADE as a Finite State Machine or FSMBehaviour and therefore can be used to model the agents in such a way that they still are autonomous but their Behaviour can still to a certain extent be controlled (see Figure 4.19).

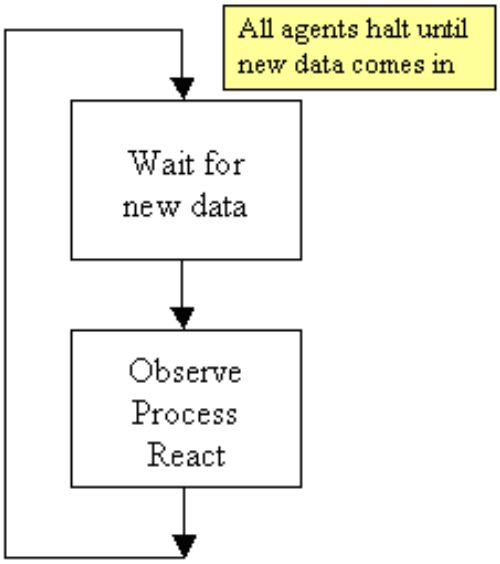


Figure 4.19: Agent Behaviour Synchronization

The FSMBehaviours used for the Agents in MACSIM are cyclic, so they keep performing the same duties in the same order during the simulation. It is only in this way that we can ensure that the agents actions will be kept synchronized during the course of the simulation.

4.4.1 SimulationAgent

The simulation agent receives requests for new Waypoints and at the same time updates the world. Via the simulation, but also via updates of the world that for instance firemen can provide. The Simulationagent keeps track of all these changes and makes sure all the ParticipatingAgents are provided with the most recent and accurate Waypoint at their particular x and y location. The central SimulationTime is also being controlled here and updated (see Figure 4.20).

WaypointUpdateNotificationHandler

The agents in the field can only see the Waypoint data in their direct environment; they only have a local copy of that Waypoint. So if an Agent, for instance a FiremanAgent decides to extinguish the fire (i.e. reducing the firelevel) then the update is only processed locally and of course not in the original data storage of the world which is controlled by the simulation Agent. The updates that are done by agents

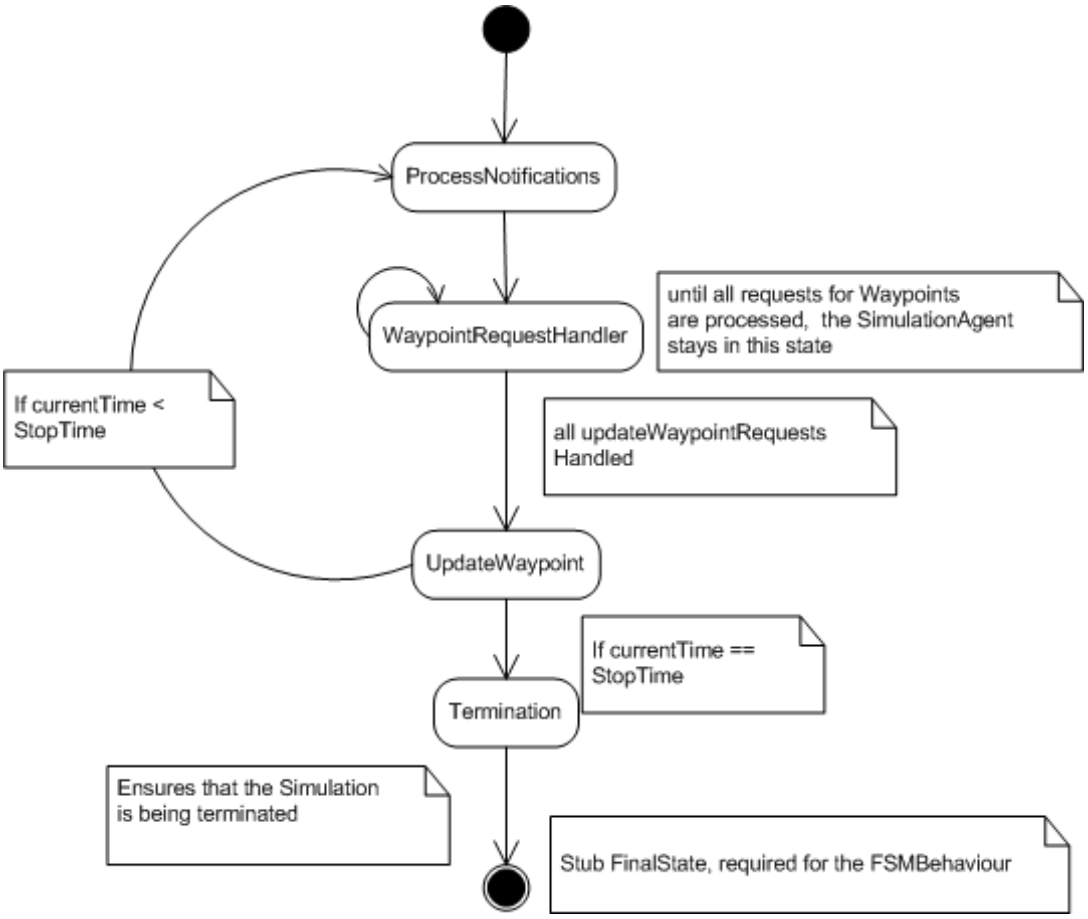


Figure 4.20: Simulation Agent Behaviour

locally therefore should be synchronized with the rest of the World. This Behaviour should synchronize update notification messages that are coming in. Based on these updates, the world can then safely be updated for the next time instance. Every agent in the field in this way receives an updated Waypoint that has been synchronized with local updates and has been processed for local time updates.

WaypointRequestHandler

All the requests that are coming in from agents in the field for information about their particular location are processed. This request basically consists of an agentCode, a request number and an x and y coordinate of which the corresponding Waypoint is needed. Based on the Waypoint grid size the Waypoint for that particular area is returned, combined with the current SimulationTime. The agent code is needed to make sure that every Agent receives the update at the same time, the request number is there to make sure that every agent receives the waypoint only once (see Figure 3.5).

This Behaviour is the counterpart of the WaypointRequestSender that is used by the ParticipatingAgents to request Waypoint updates. This Behaviour will be discussed later on.

UpdateWorldBehaviour

The updating algorithm is of course being executed by the Simulator and ScriptControl, but the command to do the actual update is being given here. This means that the central simulation time has to be updated to the next time instance (usually one minute later) and the world has to be updated.

4.4.2 ParticipatingAgent

Because the Behaviours of the ParticipatingAgents GeneralPublicAgent, Fireman, DCMRExpertAgent and ChemicalDetectorAgent have the same general functional structure, in this paragraph the Behaviours of these agents are grouped together and their respective implementational similarities and differences are treated accordingly.

These agents for now are just abstract "things" containing a position coordinate, a behaviour to observe the environment around them, and a behaviour to respond to the events around them (the basic properties of an agent), and an ID. The latter is for data transport purposes, but that will be discussed in more detail later. The only thing is that because there are different types of agents with different functions and characters. Their response and observe behaviours will therefore be different in details.

The ParticipatingAgents are part of the simulation, so in this simulation, they have to do what they were supposed to do in any environment, being agents: they should follow the agent paradigm: they observe the environment, they interpret and they react to the environment. Therefore, when the simulation has updated the environment, the Participating agents should observe, interpret and react to those changes. As described earlier, we think of this as a sort of Finite State Machine, in which every step of their routine becomes a State. For the participating agents the finite state machine looks like the state diagram shown in Figure 4.21.

First they have to get the latest waypoint data, based on their current position. When they have received that they can continue to observe and interpret the data, and then react to it. The reaction can mean all sorts of things: they can contact other agents via messaging, they can interfere in the waypoint data (firemen for instance can extinguish a fire), or they can change their movement pattern (like for instance running away or going to a certain location). When they have acquired the information, it's now up

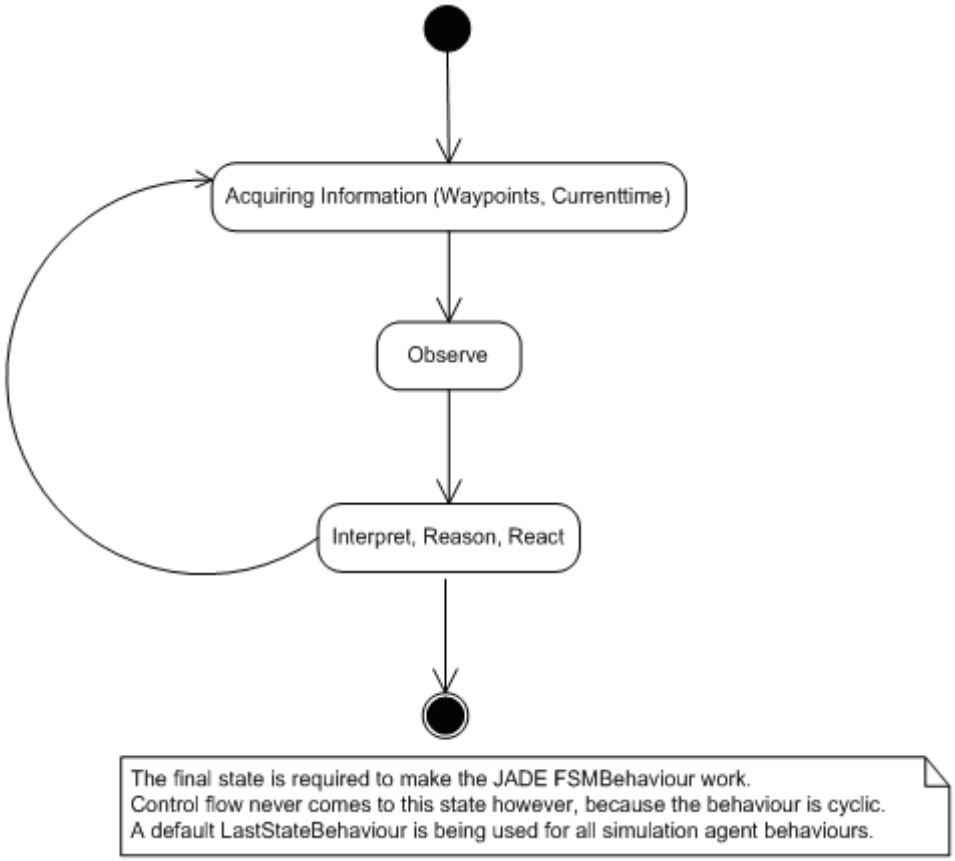


Figure 4.21: Participating Agent State Diagram

to the agents to respond to the situation. The plan is to make for each type of agent a different set of decision rules to fit to their needs and capabilities. The rules are schematically of the form given in Figure 4.22.

Of course the rules will be defined in the Jess language, which is capable of processing Java data structures as facts. We will discuss this more elaborately later in this chapter. The attributes given in Table 4.5 are the most important general attributes that every ParticipatingAgent has.

Besides having an ID, which is just a set of unique characters, they all have a data structure containing which values of the waypoint they are allowed to read. This vector is called ObservingSkills. When an agent observes the environment, he basically looks at the values in the Waypoint and only to those values that he is allowed to see. Every JADE Agent has its own set of Behaviours. We have already explained that the use cases of the Agents coincide with JADE Behaviours in the design of MACSIM. Therefore we list in Table 4.6 which Behaviours are being used by the Participating Agents.
As we can see in the table, most of the Behaviours are shared by the 4 ParticipatingAgents discussed here. We will now discuss these behaviours one by one.

WaypointUpdateRequestSender

For all ParticipatingAgents the WaypointUpdateRequestHandler has identical functionality. It sends an agent code in combination with his current x and y coordinate and a request counter to the simulationAgent, and does nothing until it receives a new Waypoint accompanied by the time it has been

```
If (visibility < 10 m )
      Moving speed = decreased
If (flash of light)
      Go to source of flash to see what
      is going on
If (current location = dangerous)
      Run away in backward direction
```

Figure 4.22: Simple Rule Examples

Table 4.5: Properties of Participating Agents

Type + Name	**Meaning**
double posX	The agent's current x-position.
double posY	The agent's current y-position.
Vector<Integer> observingSkills	The Values that a certain Agent can observe in a Waypoint. Is defined in the subclass constructor.
double movementSpeed	The speed with which the Agent can move around. For a device like the ChemicalDetectorAgent, this is of course 0 m/s.
Waypoint wayPoint	The waypoint where the Agent gets its information from.
String type	The Subclass in String format, for debug purposes.
int Agentcode	Identifier for waypoint requests.
Rete Engine	Rulebase for reasoning and storing facts.

issued in a reply message from the Simulator Agent. Obviously this reply is being created in the WaypointUdateRequestHandler Behaviour that has been discussed earlier. Both the waypoint and the current SimulationTime are being extracted, and when those two data structures are updated the agent will check whether there have been incoming messages.

AgentMessageHandler

An agent sends a message on time t, and this time will be included as a timestamp in the Reply-To slot of a JADE Message. This message is either sent in the ReactBehaviour of one of the ParticipatingAgents or in the SendMessageBehaviour of the CrisisCenterAgent.

On the receiving end, the agent will filter on the most recent timestamp, based on the current simulation time that he has received from the SimulationAgent. With this current timestamp he checks its internal messageQueue whether new messages have arrived. If so, he puts them into a local data structure for later use in the decision process. Based on a local variable in which the timestamp of recently sent messages is saved, in the same way replies to those messages in the future will be retrieved.

Table 4.6: Participating Agent Behaviors

General Public	Fireman	DCMRExpert	ChemicalDetector
WaypointUpdateRequestSender	WaypointUpdateRequestSender	WaypointUpdateRequestSender	WaypointUpdateRequestSender
AgentMessageHandler	AgentMessageHandler	AgentMessageHandler	ObserveBehaviour
ObserveBehaviour	ObserveBehaviour	ObserveBehaviour	ReactBehaviour
ReactBehaviour	ReactBehaviour	ReactBehaviour	

A message that is being sent via JADE has to be based on the JADE standards for defining ontologies. This is described in [BCTR05] in detail, but the most important thing to know is that any user defined content should always be wrapped inside a JADE object Action before it can be correctly put inside the content slot of a JADE message. This is important to know for the receiver side, because that side receives an Action object that either contains a Directive or a Report. Based on this the agent will process the message to extract the useful bits of information for storage inside the reasoning engine.

In the case a Report is received each Agent therefore needs to know what kind of events they can report about and what kind of events they can receive. Table 4.7 gives an overview of the type of events or DynamicObejcts that the agents currently can report about. A Report furthermore contains information on Time and location of this particular event.

Table 4.7: Report Content Overview

DynamicObjects	Events
Lightning	Explosion
GasSubstance	Gas Escape
Wind	Meteorological Measurement (Detector)
Air	Meteorological Measurement (Detector)
Smoke	Fire
Fire	Fire
Explosion	Explosion

Table 4.8 describes the current structure of a Directive message:

Table 4.8: Directive Content Overview

Directive Slot	Meaning
Order	What kind of order should be executed
OrderX	x-location where the Directive should be executed
OrderY	y-location where the Directive should be executed

At this point only the CrisisCenter can send a Directive. All ParticipatingAgents can send Reports, including the CrisisCenter. The crisis center only does this if it feels that a certain Report is essential for another Agent. But because not every agent can observe everything (because of their observing skills) therefore they also do not report about everything.

In the following table an overview is given of the different events that different Agents can report about.

Table 4.9: Agent Event Reporting

General Public	Fireman	DCMRExpert	ChemicalDetector
Smoke	Smoke (detailed)	Smoke	Smoke
Lightning	Lightning	Lightning	Lightning
Explosion	Explosion	Explosion	Explosion
Fire	Fire (detailed)	Fire	Fire
GasSubstance (general)	GasSubstance (detailed)	GasSubstance (detailed)	GasSubstance (detailed)
			Air
			Wind

ObserveBehaviour

For the ParticipatingAgents this behaviour is mostly the same. Based on a vector called ObservingSkills, the agent observes the values that he is allowed to read. These values are then read and stored inside the agent's knowledge base. The vector ObservingSkills is defined in the setup method of the Agent, there it's being filled with the quantities that that agent is allowed to see.

In the following table, an overview is given of what kind of Waypoint values are allowed to be seen by the different types of ParticipatingAgents.

Table 4.10: Observing Skills ParticipatingAgents

GeneralPublicAgent	FiremanAgent	DCMRExpertAgent	ChemicalDetectorAgent
Temperature	Temperature	Temperature	Temperature
Visibility	Visibility	Wind Speed	Wind Speed
Bang	Bang	Visibility	Gas Vector
Flash Light	Flash Light	Bang	Fire Level
Gas Vector	Gas Vector	Flash Light	Smoke Level
Special Objects	Special Objects	Gas Vector	Wind Direction
Fire Level	Fire Level	Special Objects	
Smoke Level	Smoke Level	Fire Level	
Wind Direction	Wind Direction	Smoke Level	
		Wind Direction	

ReactBehaviour

The most complex Behaviour of the ParticipatingAgent is the ReactBehaviour. This Behaviour includes reasoning, executing agent actions and sending agent Messages. For the reasoning and executing of agent actions the Jess rule engine and the JessEventListener are used. Before we will discuss the Behaviour itself, we will first explain these concepts in more detail.

All agents need to make decisions. This needs to be done inside this behaviour. Therefore we have to have some mechanism that is able to use predefined rules combined with constantly changing data. This should enable us to let agents make decisions that can change depending on the circumstances. One of the tools that enable us to reason inside external Java code is the rule-engine Jess. Each agent has its own reasoning engine with its own set of rules. When some changes occur in their situation, rules in their knowledge base can fire, resulting in a change of state that lead to new actions from the agents. This

can differ from for instance running away, to extinguishing fire or issuing a report to the CrisisCenter or other agents (Figure 4.23).

The Agents in the field make decisions based on the information they receive or observe. We have to

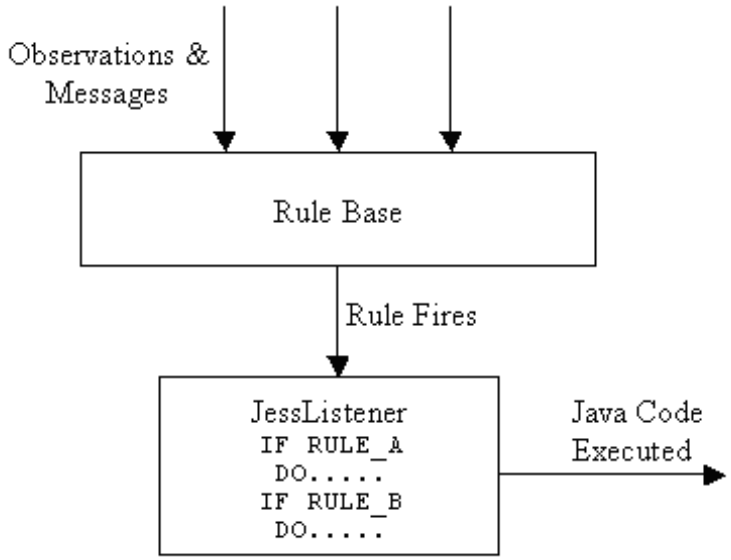


Figure 4.23: JessListener Interface

allow a flexible way of making decisions, one that is more flexible than a huge strain of if-then rules. Because if a certain condition changes through time, making decisions in regular Java code can be quite time consuming, because you have to take care of every possible situation inside Java code. Therefore creating rules in regular Java code is not a preferable option. A far better option is to include an Expert system that can process lots of user-defined rules in any particular order, which firstly makes the entire decision process more efficient and secondly the task of making rules a lot easier. Expert systems contain algorithms that process rules in a non-deterministic fashion.

The Jess Expert System Engine uses the Rete algorithm for processing rules efficiently [FH03]. Jess enables rule-based reasoning inside Java code with a special object, called the Rete engine. Instances are Rete Java objects. There can be any number of Rete objects running at the same time. Therefore it is perfect for inclusion in independent JADE Agents (which all are separate threads) that move around the world.

Every ParticipatingAgent and the CrisisCenterAgent will have their own Rete instance, which will be given a personal set of rules and the input from the Waypoint. This input will be changing all the time, and therefore the rule engine will be facing new decisions each time instance. The trick is to store the Waypoint values into the expert system as Jess facts. Whenever a Waypoint fact is still valid it is part of the reasoning engine. So each time the values of the simulation world are updated, the old fact is retracted and a new fact with new data is asserted into the knowledge engine. This assures that the knowledge engine is up to date, and when rules are fired, you can be sure that they fire based on actual live data.

It needs to be said however, that before the Jess Engine can accept a Java object as a fact, that object needs to be recognized by the rule engine. This can be done by defining a type of that particular object as a Jess-compatible fact-type. If the agents in the world all need to process those Java objects, then it might be convenient to make a dedicated Jess file that is processed by all agents that defines all the fact types. In this way we know that all the Agents can have rules that are based on the same world concepts (see Figure 4.24).

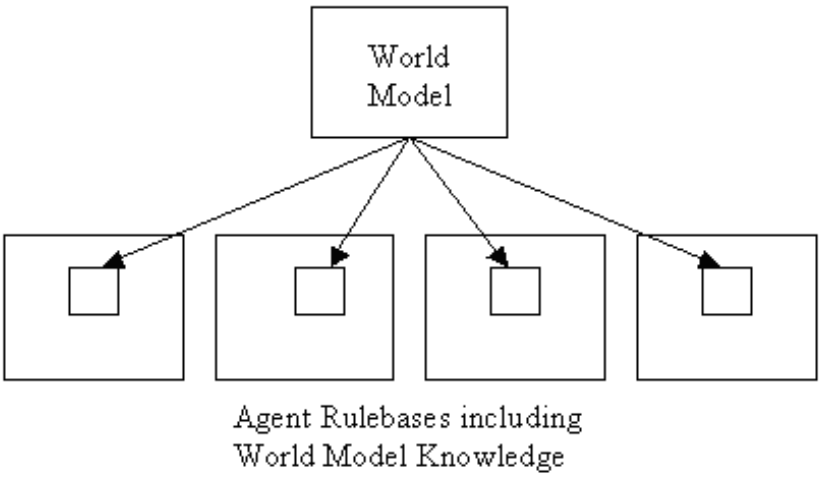


Figure 4.24: World Model Knowledge Inclusion in Agent Rulebases

Another question is how to control to the reasoning engine. When something can be concluded, the Agent should be informed of the decision and base its subsequent actions on it. The solution for this is the fact that a Rete object can launch JessEvents that can be processed by a JessEventListener. This means that via this mechanism Agent actions can be initiated based on the values of a Waypoint. The JessEventListener can listen to the engine, and in the case a rule has indeed fired this Listener can call specific actions that have to be executed. This is convenient, because in this way it is possible to execute agent actions based on observed environmental data.

Whenever an agent receives nothing that is of particular interest to him, there are no rules that are applicable for that situation. Then the agent can perform some default actions instead. This could be waiting until something actually does happen, or walking around randomly, like the General Public. The Agents will keep doing this default action until something else comes in that is important. The ReactBehaviour basically comes down to this: Because of the previous Behaviours, the Jess reasoning engine is full of observed pieces of data that are seen by the Jess engine as facts. The Jess Reasoning engine will now check facts using pattern matching. Whenever a pattern, defined in a rule is found inside the knowledge base, the rule will fire. Based on the fired rule, the agent will execute different kinds of actions. In the next table, an overview of the different possible actions is given for every agent.

Table 4.11: Participating Agent Actions

GeneralPublicAgent	**FiremanAgent**	**DCMRExpertAgent**	**ChemicalDetectorAgent**
Default: random walk	Go to Location by car	Go to Location by car	Default: full report every 5 minutes
Running away	Go to Location by foot	Reporting	Reporting
Reporting	Reporting		

4.4.3 CrisisCenterAgent

The Crisis Center is based on this global sequence diagram. First the Agent gets the data from the agents, with locations where they are reporting from. Then the messages and the locations are processed in such a way that the system can use them for reasoning. The data is stored in the InterpretedWorld

and when the system starts to reason, it will use the data inside the InterpretedWorld for generating an EventHypothesis. Based on this hypothesis, messages will be generated for the agents in the field.

TimeUpdateRequestHandler

This Behaviour is identical to the WaypointUpdateRequestHandler from the other ParticipatingAgents, but with the difference that the CrisisCenterAgent is only interested in the SimulationTime and not in a particular Waypoint. Therefore a dummy waypoint is being requested, and when the SimulationAgent receives the request in its WaypointRequestHandler, it gives in its reply that dummy Waypoint, but the CrisisCenterAgent will not do anything with it. It just extracts the SimulationTime object from the reply.

AgentMessageHandler

The AgentMessageHandler is mostly the same compared to the one used for the other Participating agents, the only big difference is that this version only scans for reports and complaints, because it cannot receive Directives. If fact, the CrisisCenterAgent is the only Agent that is able to send Directives, so from this perspective it is completely logical that the Crisis Center only expects reports from Firemen, DCMRExpertAgents or Chemical Detectors or complaints from the General Public.

WorldInterpretationBehaviour

Because the CrisisCenter has the main responsibility in times of a crisis, they must have some sort of idea what is going on in the world. This idea of what is going on in the world is based upon the status update messages they receive from the Participating Agents. This, however, does not need to be an accurate interpretation of the world. If they only receive information about a little part of the world, because the participating agents are only located there, the image of the world is based solely on those reports and no information is included about other parts of the world. Therefore the Crisis Center needs a special world, that is, not equal to the real world, but more an interpretation by the crisis center of the real world. Therefore for the WorldInterpretationBehaviour the CrisisCenter needs an InterpretedWorld.

The interpreted world is meant to be as flexible as possible. The data formats that can be added to it should be easy to modify. The reason for this is mainly because in the future, other tools may want to use the AI part of the crisis Simulator for visualization purposes and you don't want to explicitly hardcode every single thing that agents may want to report about inside the Java code. Also, you don't want to hardcode the information that can be shown on the Interpreted world (not all of the information might be coming from the agents: maybe the CrisisCenter has other sources of information at its disposal, such as geographical information, residential information and environmental information).

The interpreted world therefore consists of several layers (see Figure 4.25) that contain different types of information, and in each layer, the information does not need to be complete. If that layer consists of only three bits of information for the entire map, that should be possible. Also the number of layers in this world can be adapted. This gives the user maximum flexibility in deciding what kind of content needs to be shown inside the Interpreted World.

For now the design of the interpreted world consists of 3 layers, Layer 0, Layer 1, and Layer 2 (see Figure 4.25). The division between layers is being made because of the different types of data that are being stored inside of those layers. Layer 0 is especially for GIS data, Layer 1 is for observations made by participating agents in the real world, Layer 2 is for interpretations made by the CrisisCenter for what's going on, based on the observations in layers 0 and 1.

In the GIS layer information is stored about geographical data that is available for the crisis center before the simulation starts. Generally, a crisis center knows the area it's monitoring and therefore knows

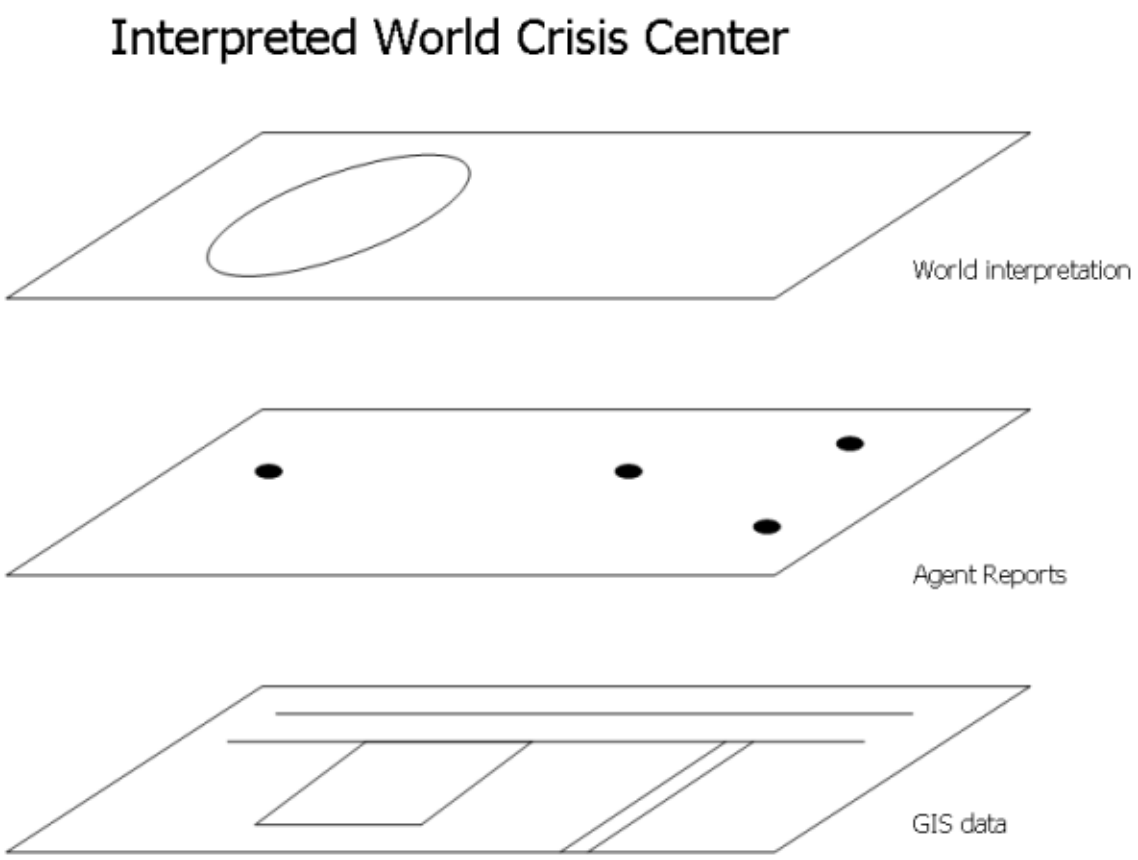


Figure 4.25: Interpreted World Design

where buildings are, where residential areas etc. This information is stored in a certain data structure, with entries for each location where GIS information is available. This should be stored in an object in such a way that 2 dimensional objects can be placed. Generally for GIS data, for buildings or specified areas (commercial/industrial/residential) rectangular space must be described, in which areas can overlap each other. For instance, inside a commercial zone there could be a very important building that is indicated separately. In the first prototype of MACSIM this will not yet be available, but in the design it is being planned as a certain future addition.

In the Agent Reports Layer the most recent observations from the world are being saved. These observations are derived from ACLMessages sent by ParticipatingAgents in the simulation. The observations are ordered by location (x, y) from where they were sent, and when additional information becomes available that can be added to the information that is already stored in the layer about that location.

Based on the information in Layer 1, the expert system will try to determine what kind of situation might be at hand and will project that in Layer 2, and (if necessary) send messages to ParticipatingAgents that are around to indicate what measurements are to be taken (This of course holds for all agents except the GeneralPublicAgent). In the message layer, the sender of the information is being saved. If the CrisisCenter makes his decisions based on beliefs or priorities, then is entirely up to the crisiscenter whether he uses this information or not. For now we do not use any beliefs, priorities of other schemes in reasoning, but if that's necessary, we only need to make changes in the decision rules that are used in the crisis center, not in Layer 1.

The World Interpretation Layer (Layer 2) interprets the situation as it's developing. This is of course an interpretation of what is going on and not the real world. It could change through time and become more accurate as more information becomes available. It contains information on the type of incident,

the shape of the area concerned. At this point this layer is not yet in use, but later on, the InterpretedEvent that is used during determining of the crisis Hypothesis could be stored in here, because it is exactly the data type that is needed for this layer. The InterpretedEvent that is used for forming the hypothesis is being discussed in the next paragraphs.

The crisis center has a rule engine which is able to add and remove the incoming messages inside it and can process those messages based on rules that are defined in advance. This represents the a priori knowledge that is available to the experts at the crisis center. As in real life, they can process messages that are coming in and evaluate their content. In our simulation this is done by comparing them against the predefined rules, in the way this is always done in an expert system. Each value that they receive is being checked against a set of rules, applied to different kinds of events and locations. When something occurs, the rules indicate which events are more probable. The rules reward "credit points" to events that are more probable. When all the messages are processed, some events have more credit points than other events. The event which has the most credit points is the one the rule base will choose as "most probable" event. Just like that, based on wind directions and XY sources from messages, the expert system will try to determine a possible source for the event. This is also done based on rules that are defined inside the rule base (see Figure 4.26).

To enable a certain theory to stand out, i.e. to really create a difference between the credibility of different probable events, the knowledge base of the crisis center must be filled with approximately 25 rules per different type of situation. Also the crisis center should contain several rules that concern determining the potential location of the incident.

When all the messages are processed, the best hypothesis is taken out and presented as the working theory. Based on that working theory, the rule engine will initiate the actions that are associated with that hypothetical event. It will inform Agents about it and instruct them accordingly. Most of the time this will mean that Directives or Reports will be created for the Agents in the field. When those messages are created, they are stored for sending to the other agents. This will be done in the SendMessageBehaviour, which will be discussed next.

When a Hypothesis has been generated, the hypothesis and the incoming messages will be visualized on a GUI. In the section GUI Design we will describe the outline of the GUI that will be used for preliminary testing and the software components that are going to be used for it.

SendMessageBehaviour

The messages that were created as a result of the fired rules are being sent here to the appropriate agents. The main reason this has been split from the WorldInterpretationBehaviour is to make the Behavior structure in the Crisis Center more intuitive. Because the WorldInterpretationBehaviour already is a very complex Behaviour in which a lot of functionality has been put, it was thought that a clearly distinct action as sending messages should better be put in another Behaviour.

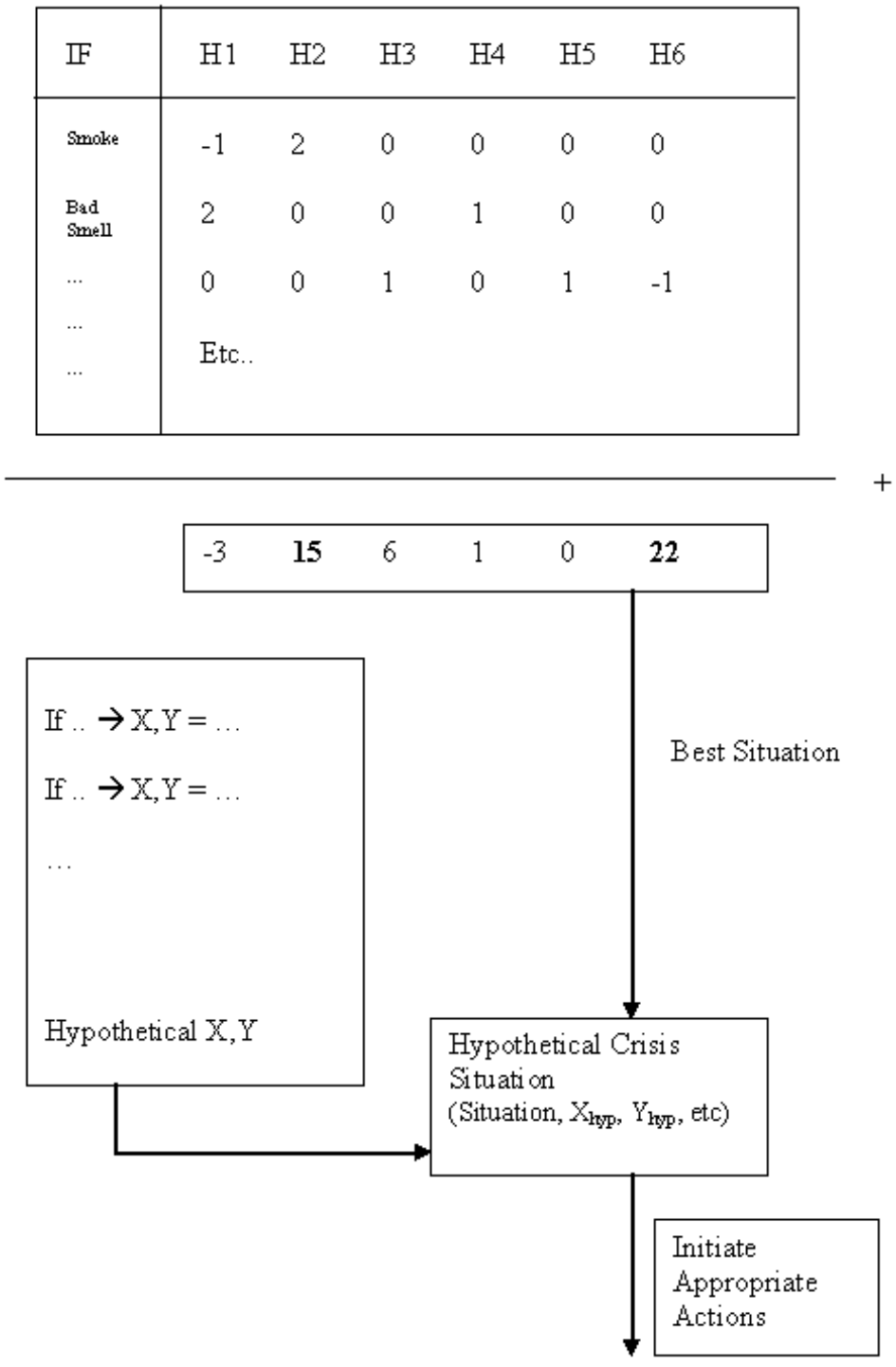


Figure 4.26: Crisis Center Hypothesis Forming

Chapter 5

Implementation

In this chapter we will discuss the implementation of the product as it currently exists. First we will discuss the differences between the design that was planned and the eventual implementation. After that we will give a short description of the tools we used for making the product as it is so far. In the final section we will focus on the way a user can currently use the system as it is right now. For instance, examples will be given of data structures, rules and scenarios and how the user can currently manipulate those data structures.

5.1 Current Status

The most important thing to remember is that the development process of MACSIM was incremental in its approach. Requirements analysis with stakeholders had already shown that MACSIM could become a very flexible simulator of crises and crisis reponse efforts. During requirements analysis, a lot of potential functionalities and components were named. The addition of new components increased the number of possibilities and implementation time. Because of time limitations it was decided to implement the essential functionalities first. The foundations upon which MACSIM could be further developed were implemented, so a proof of concept of a Crisis Simulator with basic functionality could be presented as soon as possible. More advanced features for which the user interfaces were designed, were not yet implemented. In Figure 5.1 we try to show which components have already been implemented and to which extent. It already can be said that most of the functionalities that are left out were extra features that mostly have to do with basic IO and GUI operations that can be simply added later on in the process.

Furthermore it needs to be said that the GUIs presented in this chapter are not the definitive GUIs of the MACSIM system. They are are solely used for demonstrating the current implemented functionalities. In future implementation increments these GUIs will eventually be replaced by GUIs comparable to the GUIs shown in Appendix A.

As the picture shows, most fundamental functionalities are implemented, but some of them are only partially implemented. The components and functionalities, for which the implementation functionality is not implemented as is described in the Design chapter are shown in the picture with dashed boxes around them. Details about which parts of these components are not yet fully implemented are given in the following paragraphs.

The scenario setup is not yet available through the interface at this point, because it required a lot of GUI programming that was not considered core functionality of MACSIM to show a proof of concept. It will be one of the first things that will be added to provide a more functional prototype, but for testing purposes it was not strictly necessary. We have already provided a GUI for the scenario setup functionality, but currently the parameters in that GUI are just hard coded values, that represent default Simulation

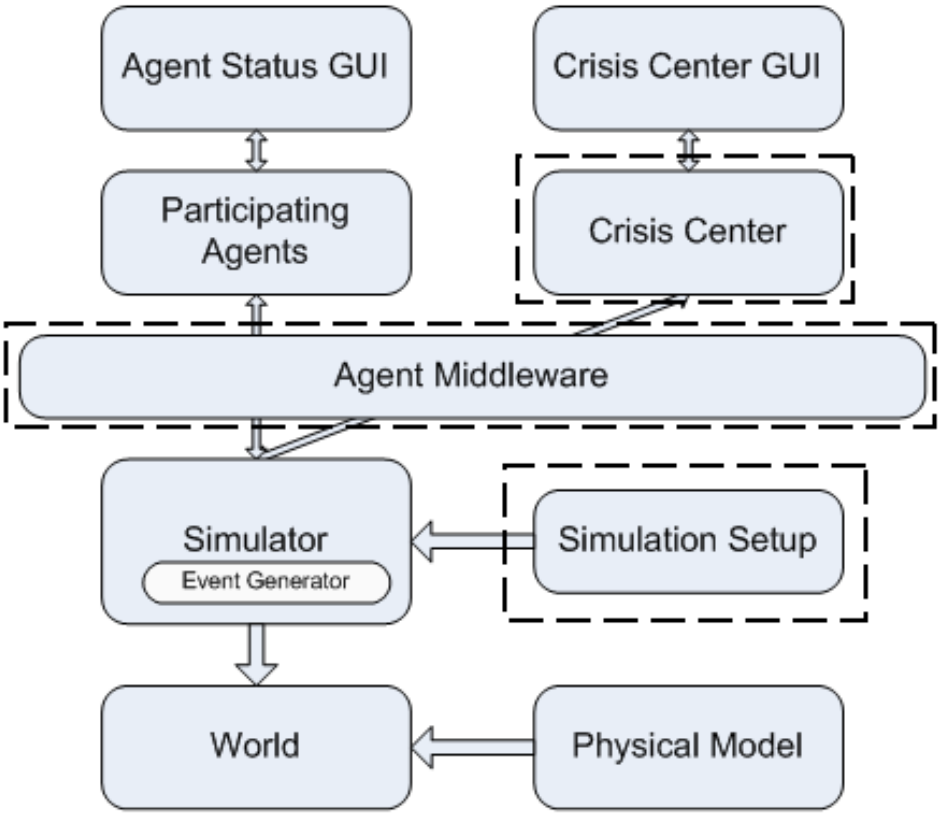


Figure 5.1: The status so far

parameters, given in the SimulationConstants and Simulation classes.

In this GUI, the user should also be able to edit the scenario that the simulation will be running. The user can add and remove events from the script that is unfolded during the simulation. For adding and removing of events, we also have prepared a GUI, but also, this is not yet functional. At this point the scenario and its events should be edited manually in the code, how this is done will be explained in the section Use of the current system. The default script is also not yet retrieved from storage but is also included in the Simulation Class.

In the InterpretedWorld, there is also space reserved for the inclusion of GIS data. As at this point there was no GIS data source available, so we left it out of the design and implementation until now. It unknown at this point if and when GIS-related features will be installed and become a part of MACSIM in a later time.

In the inter-agent communication one feature has been left out: the capability of communication between agents in the field. The JADE platform as agent middleware supports it, but at this point the agents just do not have it included in their behaviours. To prove the crisis response concept, the implementation of interaction between agents in the field and the Crisis Center was considered to be more important. If in the behaviours of Agents in the field at one point actions will be implemented to send messages to other agents in the field, the JADE framework and the Ontology used will support this, but at this point there are no such actions implemented.

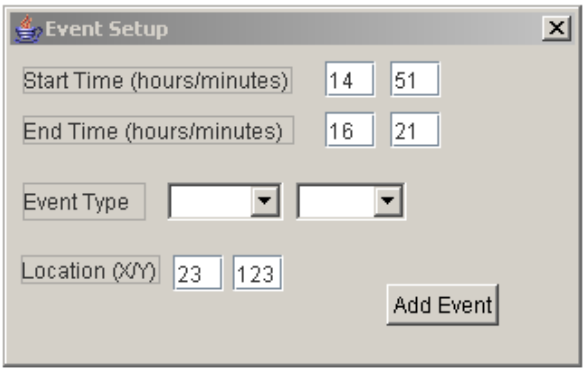


Figure 5.2: Event Setup Gui

5.2 External Tools

For the development of MACSIM, several external components were used. The particular use in the system is often already explained in the design chapters, but for completeness sake in this section a short overview is given of the external tools used during the production of MACSIM.

5.2.1 Jude

JUDE is a Java based free modeling tool for developing UML models such as sequence diagrams, class diagrams and activity diagrams and based on the object models, it is able to generate Java code templates that the programmer can start to work on. It can also be used for reverse engineering, but for that it was not used in this case, because our code contained too much Java 1.5 code, which is not yet readable in JUDE at the moment.

5.2.2 Eclipse

Eclipse is one of the most popular software design environments for Java developers, because of its open source character (it is Java-Based), and flexibility of plugin installing, it allows a lot of third party plugins to be installed. Besides the efficient Java programming features (e.g. Java 1.5 support), there therefore also is room for additional features. The additional plugins that were used for developing MACSIM were a Jess Editor plugin, which allowed very easy coding in Jess code with the same look and feel as the Eclipse user is used to for generating Java code. The other useful feature is the Visual Editor for Eclipse, which allows generating a GUI in "what you see is what you get" fashion, rather than editor-based generation.

5.2.3 JADE

JADE is the Multi-Agent platform used inside MACSIM. The fact that it uses distributed message passing plus the fact that it's Java-based makes it very attractive to use inside a Java-based project. JADE is designed in such a way that it's complying to the FIPA standards. It therefore uses ACLMessages and has most FIPA Agent communication protocols included. If the JADE binaries are included inside the Java Project, the JADE functionality can be integrated inside any Java project through code based activation and termination of the Agent platform. This way of operation is used inside MACSIM. The other alternative is to use JADE from the operating system command line as a standalone application.

5.2.4 Protégé + Beangenerator

Protégé 2.1 is an ontology creation tool. It is able to create conceptual relations between ontology concepts and based on that it can create ontology schemas in different formats, such as XML, OWL etc. When the relations are created it is also possible to generate Java code that is compatible with JADE messages passing. For this an additional plugin, called beangenerator is necessary. The beangenerator is created by Chris van Aart of Acklin. If it's installed in the Protégé directory, it is shown as an additional GUI item in the Protégé layout. It generates Java code, but it also is an option to include Java Bean functionality inside the resulting Java classes. This feature enables manipulation of these objects inside a Jess rule engine.

5.2.5 Jess

Jess is the Java expert system shell that, as it is programmed in Java, allows for an easy integration with other Java based-components. Every agent has a knowledge base, which is able to run predefined Jess rules and functions from inside Java code. These rules and functions are written in Jess language, which is a derivation of CLIPS. The Java based use of Jess can be used inside Java projects when the Jess binaries are included as project libraries in Eclipse.

5.2.6 3MNews

3MNews is a software tool developed by Iulia Tatomir at MMI lab from April to June 2006. The purpose of this program was to process Java object-messages based on the World model ontology of Siska Fitrianie in 4 different modalities: speech, an icon-map, SMS text messages and News reports. The most useful feature of this application was the fact that it can draw icons on top of a map based on Report-objects. This allowed us to only include the package called news.graphics into our system, because that contained the necessary code that was a good addition to our system. It needed to be modified just a little bit to be capable of processing multiple messages at the same time, but after that it could be used instantly. The tool generates a Jpeg picture that can be shown directly on a GUI after generation. For the generation of the picture icon files are used, thar are stored in a separate folder.

5.3 Implementation details

Specific details about the current MACSIM setup are being presented. It should be read as an overview of implemenation rationale decisions and as an explanation why certain features are currently implemented as they are.

5.3.1 Requirements & Specifications

In the current setup MACSIM can be run on a single PC as a stand-alone application. This means that all inter agent message traffic is comes down to local message passing instead of Internet messaging. This of course means that the firewall installed on the pc should know about this and should not block outgoing data traffic from the dedicated JADE (i.e. incoming and outgoing) data traffic ports. The only thing that needs to be installed is a Java runtime environment 1.5, and the required binaries and data folders of external components such as JADE, Jess and 3MNews, Eventually it is going to be a stand-alone application, which can be run from the command line. At the moment it runs within the Eclipse environment, but for the functionality this does not make any difference.

5.3.2 JADE interface

JADE is currently used as a foundation for the MACSIM program, which means that an instance of the JADE platform runs underneath the simulation. Because all data traffic takes place on a local PC, for

this first prototype the Agent ID's are hard coded inside the source. In later versions the agent names to which messages have to be sent, should be retrieved via the Directory Facilitator (DF) service while asking for the "service" that particular agents have to provide. Then the DF will provide the agents with a list of agents that are applicable to that request. For testing purposes, the system does not necessarily need this, so at the moment there are only hard coded agent names used in the code.

Because during simulation setup the properties of the simulation that is to be run need to be determined, first the Simulator is being run. In there all the necessary parameters are being set for the actual simulation to function. When this is done, the JADE environment has to be started to get the actual simulation running. The simulation gives the JADE startup command so that the simulation can run. JADE keeps on running during the simulation until the end of the simulation has been reached. When this is the case, JADE and all the other components can be shut down. The most elegant way to achieve this was to let the user shut down the system by pushing a stop button. The only thing was that the CrisisCenter, which did not have knowledge about when the simulation was actually finished, owned the GUI on which this button should be placed. This was solved by letting the SimulationAgent that does know this, inform the CrisisCenter about the end of the simulation by sending an ACLMessage. When the CrisisCenter receives this, it notifies the user about this by showing a notification on its GUI that the user can close the system without missing any upcoming updates.

5.3.3 Jess Interface

At the moment the Jess engine is called inside the dedicated AgentBehaviour through calling its run() method after gathering all the necessary data to start reasoning. At this point the Java object data is being put into the Jess rule engine, but only for pattern matching to see if certain rules apply. If certain rules apply, the consecutive actions that should be the resulting Right-Hand-Side of the firing rule are placed outside the Jess code inside the JessListener. This is to reduce the number of data operations inside the Jess engine, a data structure that is meant to pattern match as fast as possible but is not suited for many object data operations if it's not particularly necessary.

Therefore most Java object operations are carried out inside the JessListener instance that is included inside the Agent. This means for instance that the forming of the hypothesis takes place inside the JessListener Java code instead of inside the Jess engine as the Right-Hand-Side of a rule. It does exactly the same as described in the design chapter.

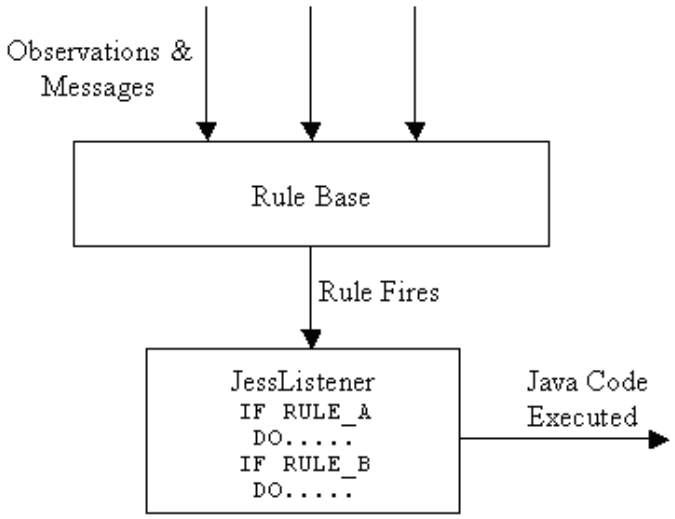


Figure 5.3: Jess Interface

5.3.4 GUI Design

The GUI currently used was made inside the Visual Editor plugin, and consists of Java Swing components. This was because of the fact that JADE Agents are supposed to have a GUI based on Swing components, because of event scheduling issues [KBR05b].

In this implementation cycle the GUI design was largely focused on testing, and showing the internal functionality and Agent message traffic. Therefore a lot of text fields are included in the current setup. In the following two figures we show the GUI as it's currently used, and describe which item is which. For a detailed description of the GUI functionality, please refer to the section Use of the current system.

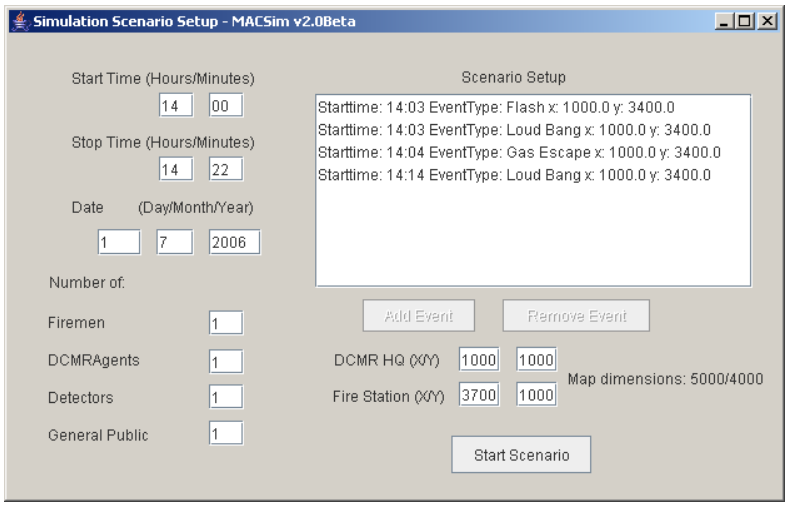


Figure 5.4: Simulation Setup Interface

As shown earlier, in the GUI shown in 5.4 the Simulation Parameters can be defined, as far as they are allowed to be changed by the user.
In Figure 5.5 the interfaces that are shown during running of a simulation. The left screen is the screen that shows which events of the script still have to take place. Whenever an event is launched it is shown in here. The middle frame is the frame that shows all incoming and outgoing communication plus the status inside the CrisisCenter. The frames on the right show the status of the agents in the field. They indicate the position in the world and describe the actions currently executed by the agent.

In the GUI of the Crisis Center, a picture is shown of the current situation of the crisis, based on the messages that are coming in. The messages are taken from the Interpreted World and from then on, with help from components from 3MNews, are represented in the form of icons on top the map. According to the design, the interpretation of the crisis is also stored in the interpreted world, for future display on top of the map. This is not yet the case at the moment. At this point the crisis is only represented as textual description on the Crisis Center GUI. There is a place reserved in the InterpretedWorld data structure, but as the components that generate the map picture are at this point only capable of placing icons on a map, this feature is not yet included.

5.4 Use of the Current System

This section is meant as a short manual for using the current version of the system. Step by step the different available functionalities are shown and explained through pictures. Elaborate examples of data structures, Jess rules and messages are given in Appendix C. The reason for this is that we tried to separate text and large pieces of Java code within this thesis as much as possible. In most cases only

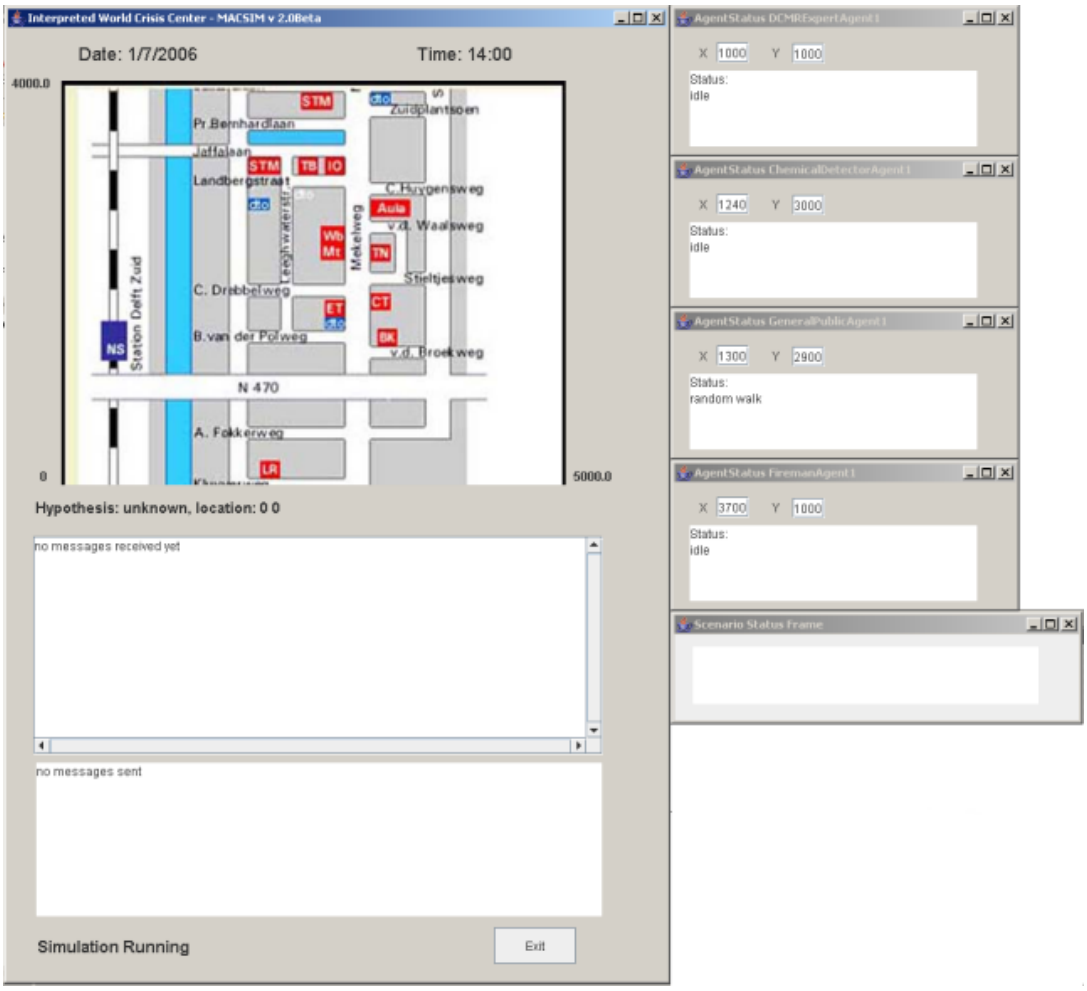


Figure 5.5: Overview MACSIM GUI Components

the results of the functionality that is being programmed is shown in this section. For a more detailed explanation of the currently implemented algorithms we refer to chapter 4.

5.4.1 Scenario Setup

A GUI is available for the scenario setup (see Figure 5.4), but currently this is just used to show the hard coded scenario that is run by default. If the parameters of the Simulation have to be edited, the values currently mentioned in Scenario.java have to be changed. At the moment this are the script status, the number of agents that are participating and the location of fire department and DCMR Headquarters. Also of course the start and end date and time of the simulation are defined here.

Other data that is necessary for the simulation like the World size but also some other global constants like default Waypoint values are described here. These values are not to be changed inside the GUI, but can be changed inside the class SimulationConstants.java, which is added in the appendix, to get an overview. Also the events are hard coded at this point, so they currently cannot be changed at runtime. They are also defined inside Scenario.java and must be added manually inside this class. Later on this will be done by the GUI presented in the previous section. For examples of adding events, see the appendix. When the user has checked all parameters, he can press the start button and the system starts the Simulation.

5.4.2 Running of the Current System

We now see the GUI's from the last section in action. All the possible events that are currently implemented inside MACSIM are shown in one or more of the GUI screens shown earlier. The time and the date are currently shown on the screen and updated during each time instance.

For each event that can take place inside the system, we will now describe what the user currently sees of this.

Event Launched

To keep track of the events that are launched, the launching of an event is shown inside this GUI:

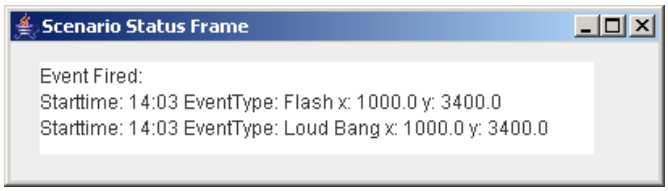


Figure 5.6: Event Launched

Agent observes something

If the agent observes something of interest (Figure 5.7), it is shown inside its status frame in the text field, it is then sent to the crisis center, where it will appear in the list of incoming messages and a new icon will appear on the map from the location from where the message has been sent (see Figure 5.8). This will only be seen in the next time instance, because it takes time for the crisis center to process the incoming messages. The agents can observe and report all kinds of things, depending on their function in the simulation. Examples of this will be shown in the evaluation chapter, where two scenarios will be shown, with all different types of observations made by different agents. Also actions are shown indside this window.

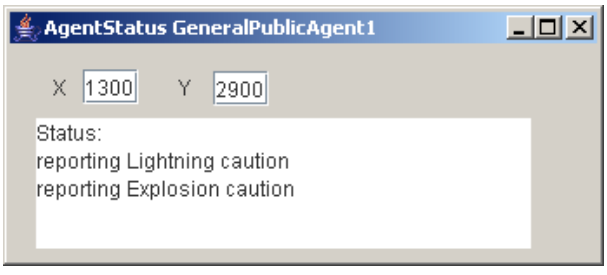


Figure 5.7: Agent Observing in GUI

Hypothesis forming

When the simulation starts, the crisis center does not know what is going on. Therefore it shows on the GUI that the possible event and location are unknown. When messages start to come in, this starts to change. The maeesages are shown on a map and it eventually becomes more clear for the Crisis Center what is going on, because the theory and the possible locations are being updated every time instance, based on the message information available at the Crisis Center. Because every time instance

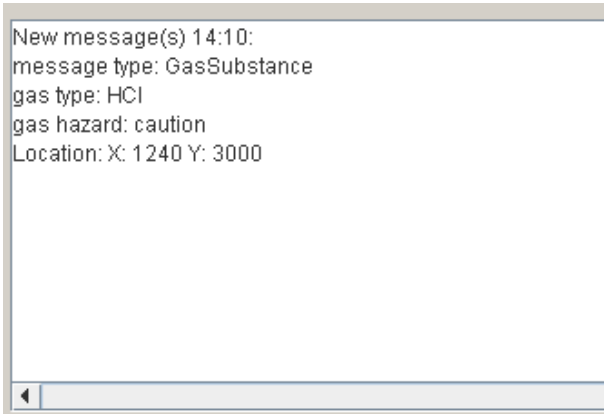


Figure 5.8: Incoming Messages at the Crisis Center

new messages can come in, every time instance the hypothesis and the location are subject to change (Figures 5.9 to 5.11).

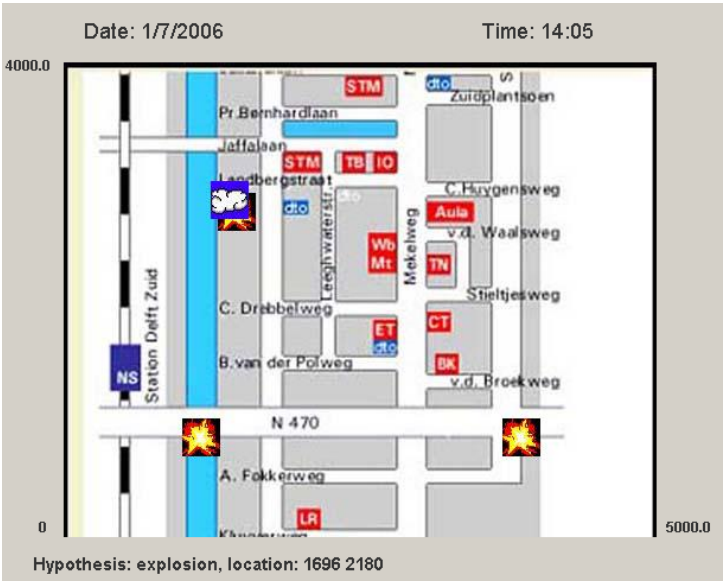


Figure 5.9: Crisis Center Map

Figure 5.10: Hypothesis Creation: Unknown

Crisis Center responds

At a certain point the Crisis Center becomes convinced that a certain crisis situation is going on. It then mobilizes either the DCMR Expert or the Fireman to go to the location where a certain event might be going on. The recipient then gets this message and goes to the desired location. This can be seen by its location parameters that are updated and the status change of that agent (Figures 5.12 and 5.13).

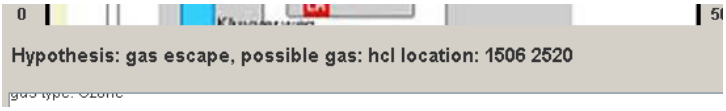


Figure 5.11: Hypothesis Cretation: Formed

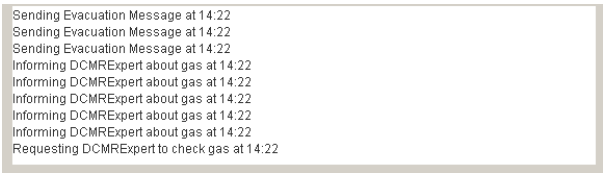


Figure 5.12: Crisis Center Responses

Expert asks for Evacuation

When the Agent has arrived at the location, it determines what is going on. When it considers the observed situation as unsafe for the general public it issues an order to the Crisis Center to evacuate the General public. The crisis center does this by sending a signal to the general public to go away from the possible disaster location. The agents then run away if they are close to that area (Figure 5.14).

Simulation finishing

When the simulation is finished this is shown by a label in the lower left corner. Now the user can push the exit button and the program shuts down (Figure 5.15).

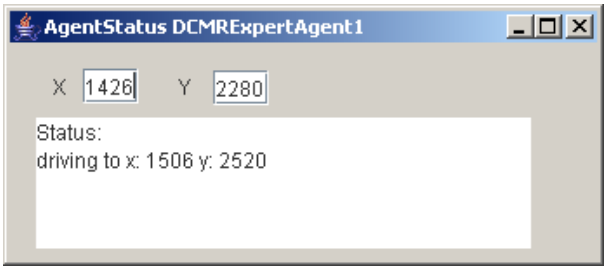


Figure 5.13: Expert going to a Location

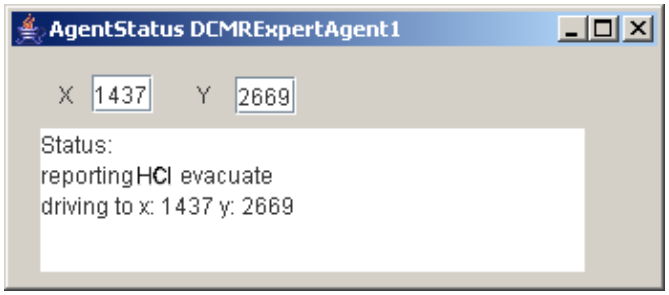


Figure 5.14: Expert Orders Evacuation

Figure 5.15: Simulation Finished

Chapter 6

Evaluation

In the previous chapters we have described how we conceived, designed and implemented a Crisis Simulator, Interpreter and Monitor. In this chapter we show how we tested it. The main criteria of a Crisis Simulator are of course that it should be able to simulate a crisis and the actors in the crisis area respond as expected. We tried to test this by making up two test scenarios that could happen, and in which all the aspects that we have modeled inside MACSIM are incorporated. The main goal of the evaluation was to find out whether the scenarios could be represented in the current world model and if the agents would behave as expected.

The scenarios were made with the main questions of this thesis in mind (see section 1.2). Therefore we tried to make the scenarios in such a way that everything that can be simulated is actually simulated (question 1), it therefore uses all functionality of the simulator plus event generation possibilities currently available, with waypoint updating and model unfolding (question 2),it contained simulation data that is as realistic as possible (question 3), it fully utilizes the available multi-agent communication functionality (question 4), and the participating agents AI will be tested for determining the crisis event and location (in the case of the crisis center) and for real-time environment interpretation (in the case of the agents in the field). This is related to the fourth sub question of this thesis. With the two test scenarios addressing the four sub questions of our thesis, we hope that the outcome of this evaluation can give an answer to the problem definition posed at the end of 1.

The hardware and software used during the evaluation is the material on which MACSIM is currently running and is described in the implementation chapter. The test scenarios were each run on a separate day. To optimize the results of the outcome the simulations were run multiple times. The simulations were implemented inside MACSIM as described in the appendix. This is because at this point the scenarios can not yet be described in files or storage. But the scenarios can be programmed as they are described in the following section; only the input format at this point is not yet user-friendly.

For the creation of the scenarios we were forced to make some assumptions to keep the information flow controllable. Otherwise the crisis center will be receiving a lot of messages which for testing purposes makes it not easy to follow if the basic principles of the system are working. Also the agents for the scenario need to be in a certain position to pick up information, because otherwise the scenario would take too long. In the next paragraphs we discuss the assumptions made for the scenarios.

Agents sometimes are on fixed positions. For instance, the general public is supposed to be walking around in a random fashion, but to make sure that the agent is at the right position to pick up the waypoint values that he should be picking up at a certain point in time; it is default standing still on a certain predefined position. This does not mean that the General Public cannot move around. If the waypoint values or messages from the Crisis Center require it to run away it certainly can do that, but for receiving waypoint values that allow it to display such behavior, it was inevitable to fix their position. The only reason this was done was for testing purposes.

Only one example per type of agent is located in the world at a time to prevent an overflow of agent messaging. To monitor the behavior of all the currently implemented agents, it was better to have only one agent per type at once. This was also more efficient for tracking the messages that those agents would be sending to the Crisis Center.

The disasters were placed in the world at such locations that the effects of a disaster could be demonstrated as optimal as possible. This means that the crisis first unfolds at a location where only the general public and chemical detectors can read extreme values. The Fireman and DCMRExpert are placed at a greater distance from the crisis source because they at first need not be involved, only when the Crisis Center wants them to be involved they are mobilized. Furthermore the events all happen at the (roughly) the same location. This is mainly because of the limitation of one crisis hypothesis at a time. If there are two different events at two distinct locations, more different crisis hypotheses will be necessary and the decision rules should be able to cope with this. At this point this is not yet the case, so multiple crises at different location are currently left out of the testing scope.

The length of the scenario is purposely shortened because only the phases of crisis response that currently can be shown have to be part of it. Because dynamic scripting and waypoint changes by agents are currently not implemented it was not considered useful to stretch the length of the scenario to the point in which crisis response phases, that require these features, would have to be included.

6.1 Evaluation Scenarios

As we have said earlier, we have developed two test scenarios for evaluating the models algorithms and assumptions currently available in the prototype. We will now describe both scenarios, show step by step the effects of that particular scenario in the current implementation and then discuss things that work out fine and details that leave room for improvement.

In the following two sections we will first describe the starting situation of the world and the agents. Both scenarios are then written out as short stories. These stories are an idealized form of the scenario actually represented inside MACSIM, because certain elements of these stories are cannot yet implemented in the system as it is right now, but it illustrates what kind of scenarios could be tested inside the MACSIM environment. The stories as such will be quite similar when they are being read, but that is because the scenarios are quite similar as well. Then we will continue to give an overview via screenshots what the effect is during execution of the scenario using MACSIM. There is a tricky balance in this. Either we would give no pictures at all, which would result in a very unclear view for the reader what happens while running the program. On the other part of the spectrum we could give a program view of all interfaces during every time instance of both simulation scenarios, wich would leave us with 44 full-size screenshots, or even more, when individual components are shown. This would not readability, so it was decided to tell what is going on while running and to only show screenshots where it is absolutely necessary to understand the program.

Some properties of both scenarios are the same. In both scenarios at the starting point it is June 1st 2006, in a world of 4 km in the y-direction and 5 km in the y-direction. The origin of the coordinate system used lies in the lower left corner of the map. The coordinates used in the simulations therefore are relative coordinates to the lower left corner of the world. The temperature in the world is 20 degrees Celsius and, the wind speed is 1 meter per second.

There is a fire department and a DCMR Headquarters located in the world, where one Fireman and one DCMR Expert respectively are located. The Crisis Center is not located inside the area currently shown on the world map. At the beginning of the scenario they are both idle, waiting for something to happen. One member of the general public is always walking around randomly (although in this scenario it is standing still because of the reasons explain in the previous section) and one Chemical detector is placed

inside the world. It gives a status update about weather conditions every 5 minutes.

6.1.1 Scenario 1: Fire

The starting position of the world is as follows:

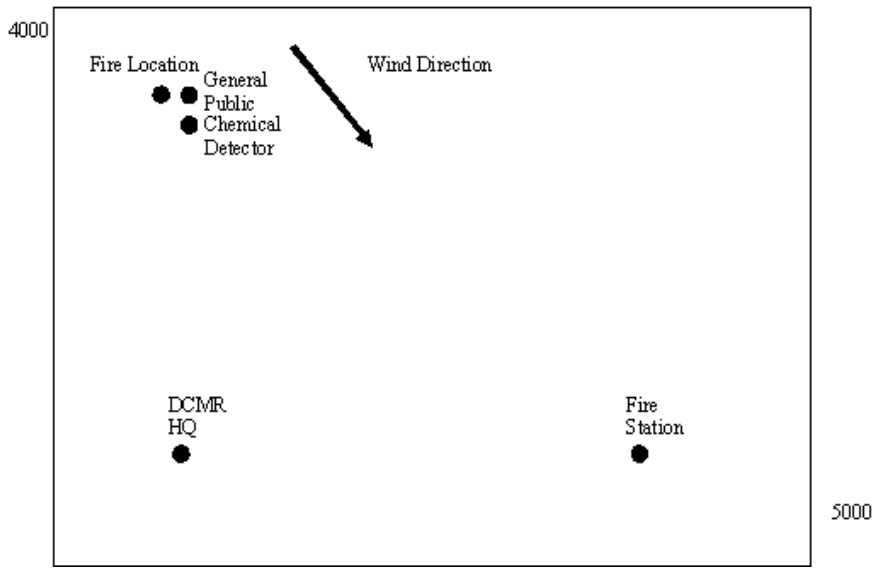


Figure 6.1: Map of the Fire Incident Area

Scenario Story

At 14:00, all is quiet in the world. At 14:03, a huge explosion is taking place, and at the Crisis Center messages are coming in. Reports of a big bang and a flashlight are coming in from multiple sources. The Crisis Center suspects that an explosion has been taking place, but feels it doesn't have enough evidence yet. Therefore it stays on high alert while waiting for more information from the field.

On 14:05, a report is coming in from a detector in the area of (LOCATION) that a fire has broken out. on 14:06, members of the general public are calling that they saw a fire at (LOCATION) and ran away. This is enough credible evidence for the crisis center to send a unit of Firemen to (LOCATION). It sends a report with all the available information to the fire department. The fire department reads this and at 14:07 a unit of Firemen is on its way to (LOCATION). While they are on their way, after a couple of minutes, they hear another explosion. At 14:12 later the unit of firemen arrives at (LOCATION) and they see that a large fire has broken out with a range of approximately 80 meters. They report back to the Crisis Center about it and give the advice to evacuate the area directly surrounding the crisis location. In the meantime the crisis center received regular alerts from the chemical detector reporting about smoke and fire at (Location). The Crisis Center issues an evacuation signal per SMS to the general public to leave the area surrounding (LOCATION). At the same time the Firemen start to extinguish the fire. The scenario ends at this point.

Scenario inside MACSIM

We tried to model this story into a simulation that could be shown inside the MACSIM environment. At every minute that something happens inside the GUI a screenshot is being made. Through highlighting the important outputs we try to explain in the next section what the user can currently see running this scenario.

At first, when the scenario has just started (at 14:00) nothing happens, the agents are doing nothing, the crisis center doens't know anything about about a disaster. We can see this because there is nothing happening on any of the status frames.

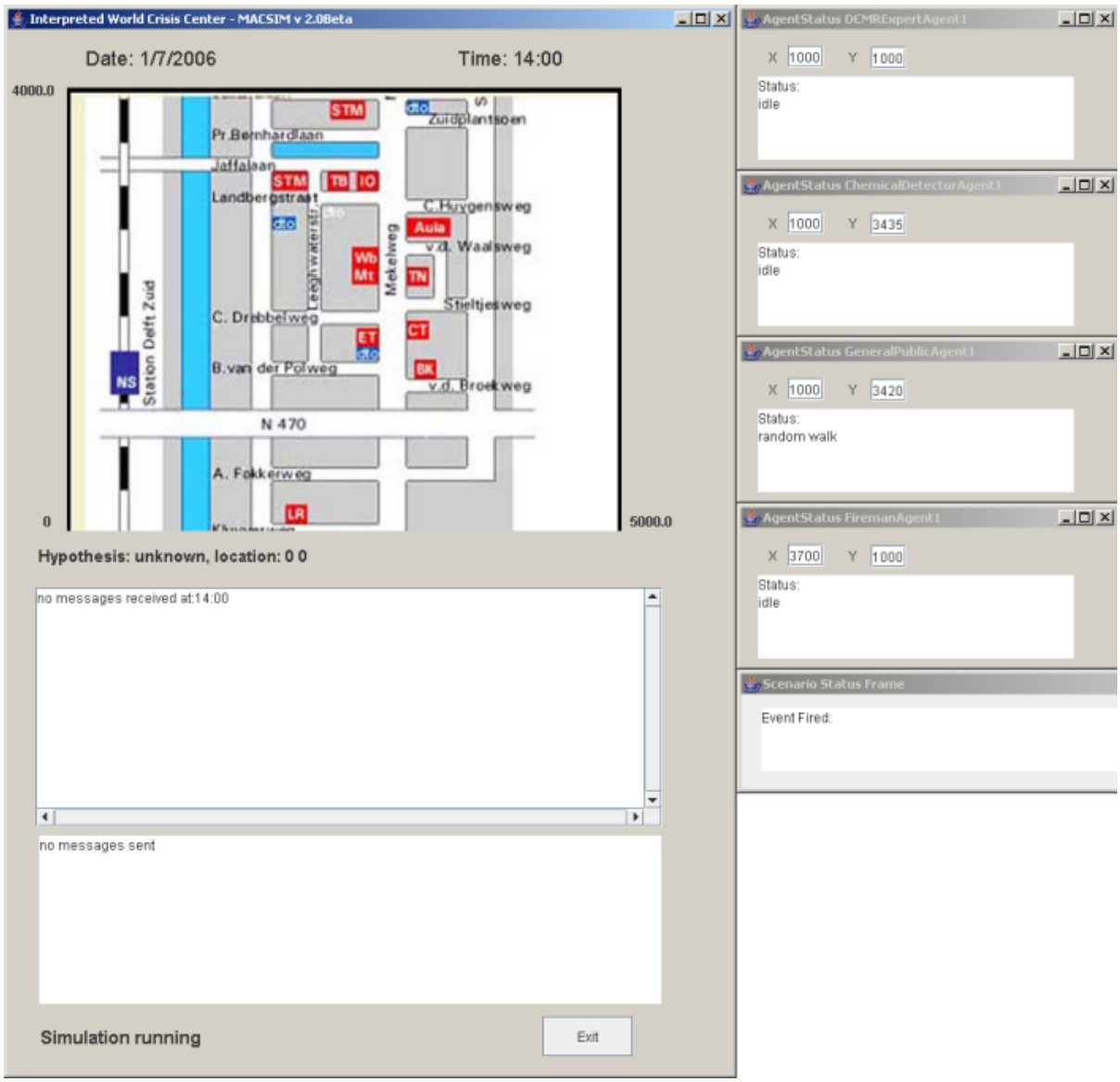


Figure 6.2: Fire Scenario: Nothing Happened

Then suddenly on 14:03 a flash and a bang can be heard. The scenario frame shows this, and gives the location of the incident. The agents are observing this, and they report about it to the crisis center. This will be shown a minute later in the lists of incoming messages and new icons on the crisis map.

The hypothesis will be updated as well, because the crisis center has now messages to generate a hypothesis in the first place. It changes from 'unknown' to explosion. The only thing is that the possible location is not right, because the crisis center currently takes the average of all locations mentioned in the messages. Because the flash and the bang can be seen from any location all agents that can hear about it report about it. Therefore from all locations where are agents that can hear it the average is produced.

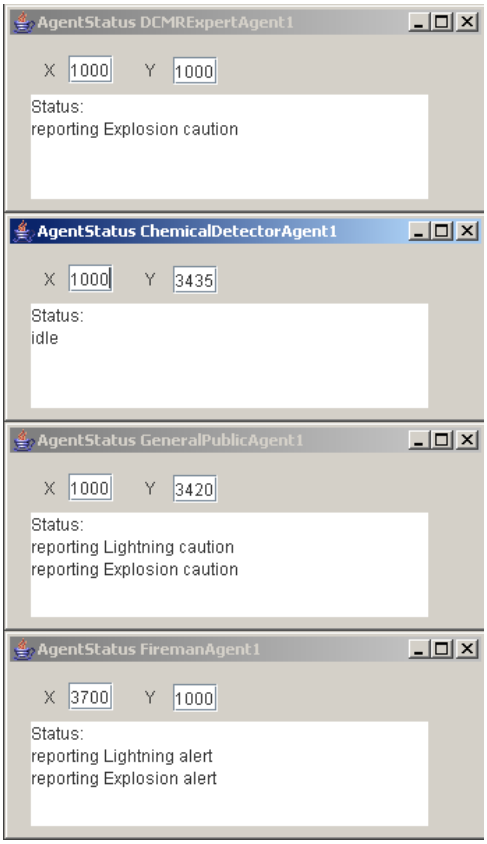


Figure 6.3: Fire Scenario: Agents observe Explosion

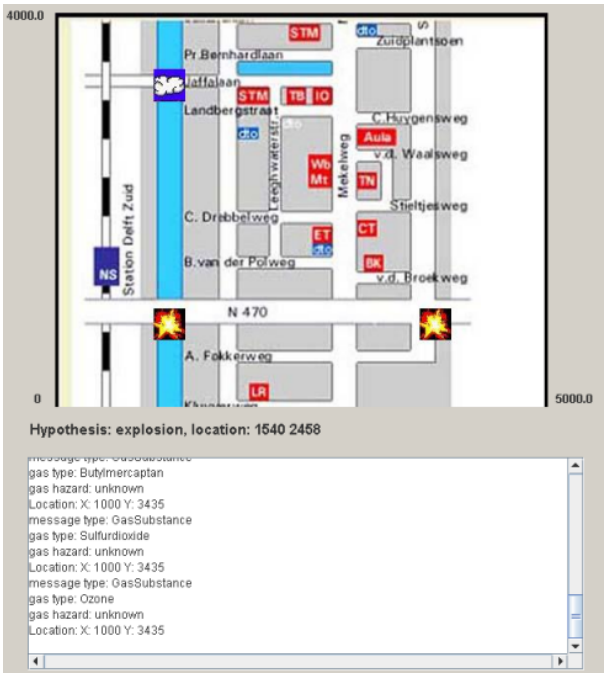


Figure 6.4: Fire Scenario: Crisis Center receives first Messages

Meanwhile we see another event developing. A fire event is launched. In the following minutes the fire will be spreading and the General Public and and the Chemical Detector will receive alarming rates

Figure 6.5: Fire Scenario: Hypothesis Explosion

of fire and smoke. They will observe it and send a message about it to the crisis center. When this message is received by the crisis center, fire icons appear on the map. The hypothesis starts to change into another hypothesis, being a fire. Unfortunately the location is based on the average of all message locations (including the old ones reporting about explosions). Therefore the hypothetical location is not close to the source of the incident. Eventually, during the simulation, the location will converge to the real incident location as more and more incident reports start to come in. The general public, as the see fire, report about it only once and run away, so the location will change instantly.

Figure 6.6: Fire Event Launched

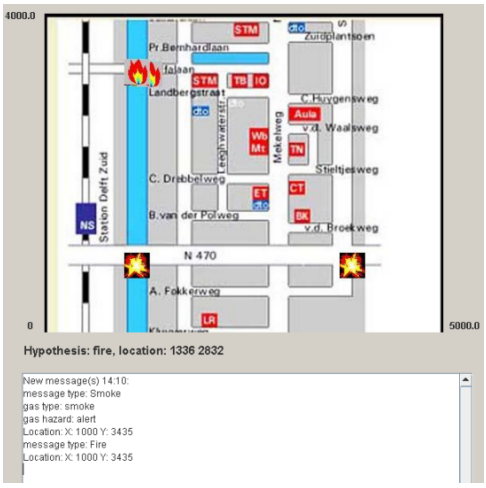


Figure 6.7: Fire Scenario: Crisis Center receives first Fire Messages

Figure 6.8: Fire Scenario: General Public running away

The crisis center then orders the fireman to go to the location where it thinks the fire is going on. Unfortunately, the fireman goes to that location by car but does not observe anything, as there is obviously no fire there. As more information about the fire comes in, the fireman comes closer and closer to the right location, but this still is not good enough. At 14:22, the simulation is stopped and the user can click on the exit button to terminate the program.

6.1.2 Scenario 2: 15000 kilograms HCl gas escape

The starting position of the world is as follows:

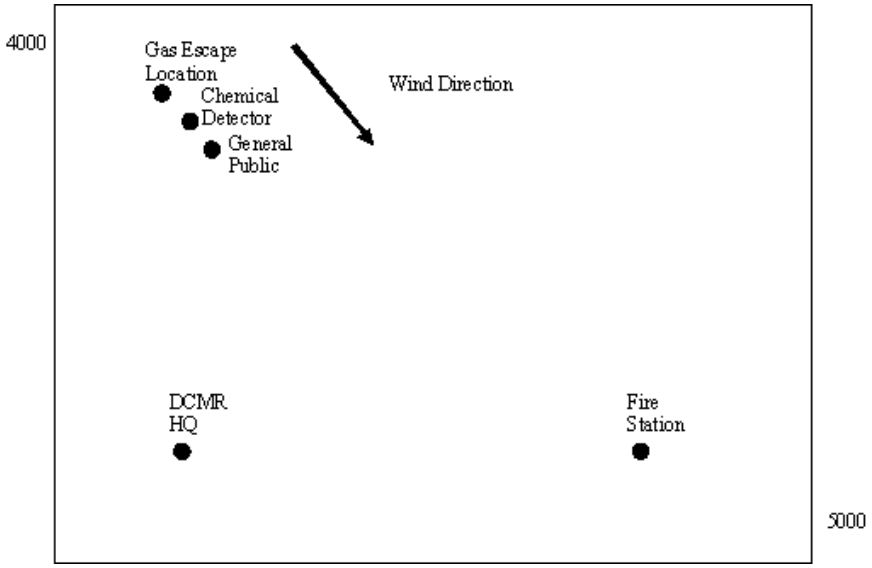


Figure 6.9: Map of the Gas Incident Area

Scenario Story

At 14:00, all is quiet in the world. At 14:03, a huge explosion is taking place, and at the Crisis Center messages are coming in. Reports of a big bang and a flashlight are coming in from multiple sources. The Crisis Center suspects that an explosion has been taking place, but feels it doesn't have enough evidence yet. Therefore it stays on high alert while waiting for more information from the field.

At 14:08 complaints from the general public start to come in about a sore throat and a chemical odor. Also the chemical detector gives worrying readings for measurements for HCl. At a certain point the people in the Crisis Center are positively sure that a huge amount of gas is escaping and a gas cloud is moving along with the northwest wind. This is based on the information that the Crisis Center has been receiving through the general public and the chemical detector in the area. Based on the complaints and the readings from the chemical detector the Crisis center suspects that the escaped gas might be HCl. Therefore the crisis center orders the DCMR Expert to take a look at the supposed location. The DCMR Expert gets underway and while he is underway, he hears more bangs coming from the supposed crisis location. When he arrives at the location, wearing his protective suit, he gets extremely high readings for HCl. He immediately gives the order to evacuate to the Crisis Center. The crisis center sends this order through SMS to all the members of the general public. The scenario ends at this point.

Scenario inside MACSIM

In this scenario, the first couple of minutes are mostly the same as the first discussed scenario. Therefore some of the descriptions used are the same as in the description of the other scenario.

At first, when the scenario has just started (at 14:00) nothing happens, the agents are doing nothing, the crisis center doens't know anything about about a disaster. We can see this because there is nothing happening on any of the status frames. Then suddenly on 14:03 a flash and a bang can be heard. The scenario frame shows this, and gives the location of the incident. The agents are observing this, and they report about it to the crisis center. This will be shown a minute later in the lists of incoming messages and new icons on the crisis map (see pictures in previous section).

At 14:04 gas is escaping, but at first the observers do not notice this, because the concentrations at this point are negligible. Later on, the agents start to notice something and they issue complaints.

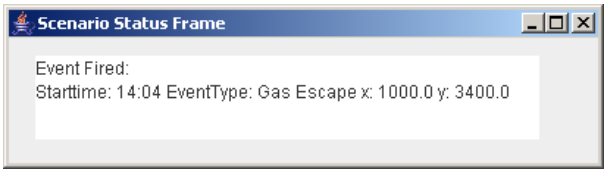


Figure 6.10: Gas Scenario: Gas Event Launched

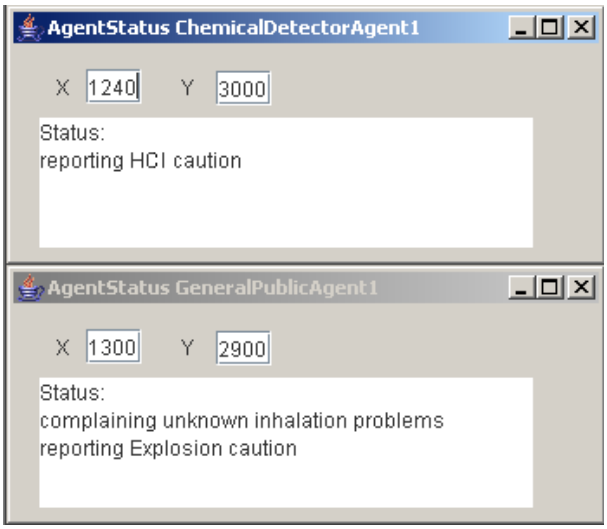


Figure 6.11: Gas Scenario: Gas observed by Agents

Later on the crisis center learns more and more about the situation and based on the incoming messages it finds out that a gas escape might be at hand, and the type of gas is also determined based on the observations. When the crisis center is certain that a gas escape is happening, it orders the DCMR to check it out. As was the case with the fireman, the location determination is not accurate, but as we will see, in this case it is not a problem. Meanwhile, the DCMR Expert receives new orders with a new location that lies closer to the real source, because of the same reasons as mentioned in the previous scenario.

Based on the measurements, the DCMR expert decides that the level of HCl in the is too high and gives the advice to evacuate to the crisis center. Based on this the crisis center will send a directive to the general public to evacuate.

The crisis center starts to send more and more messages to the agents in the field. This is because the decision whether a message is being sent or not is currently based on all the messages arrived so far. So if in time more and more messages start to come in, more and more reactions are given, by the crisis center, some of them are the same.

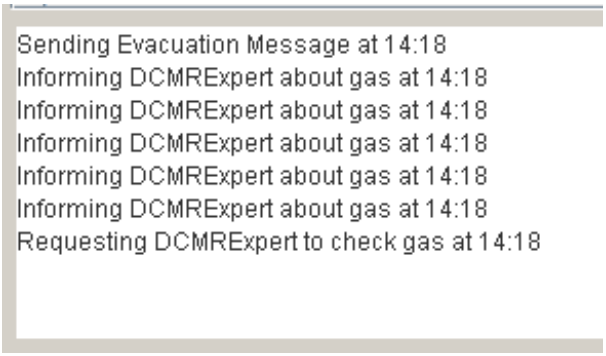


Figure 6.12: Gas Scenario: Crisis Center orders DCMR Expert

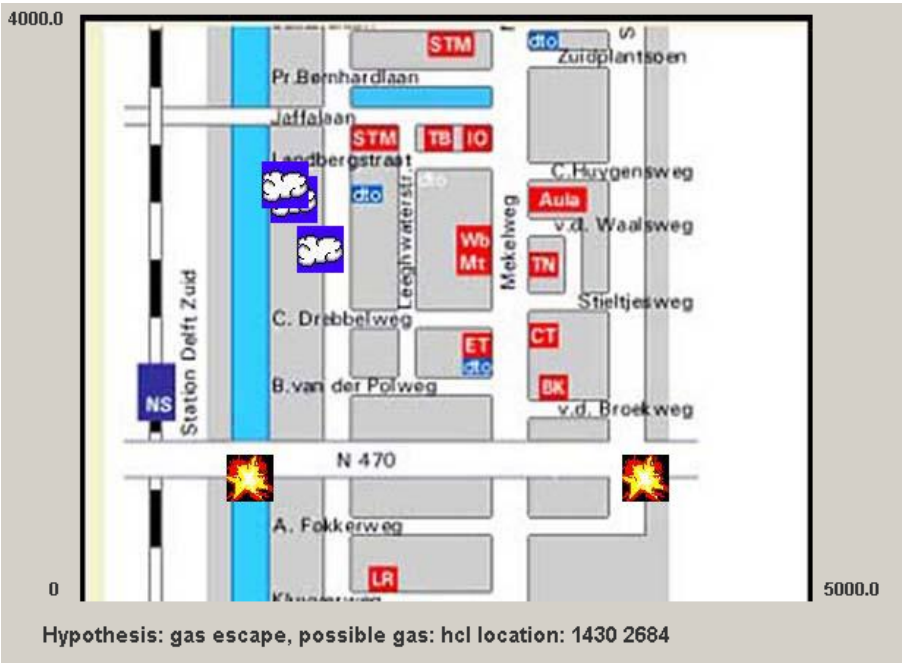


Figure 6.13: Gas Scenario: Gas Info appears on Crisis Map and in Hypothesis

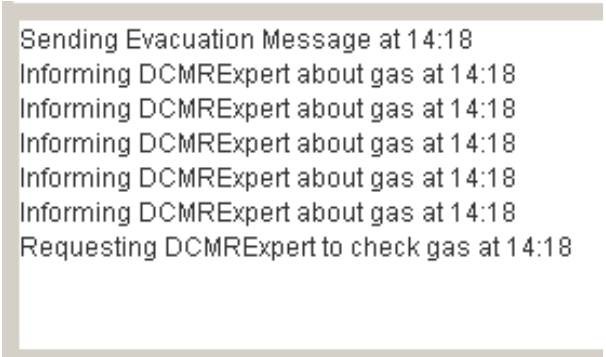


Figure 6.14: Crisis Center sends Evacuation Message

Finally, at 14:22, the scenario ends.

6.1.3 Results of Experiments

At this point the program is able to model the entire chain of communication that is required for the sending and receving of messages, interpreting those messages and giving a reponse to it. At the moment it is possible to run simple scenarios such as the examples described above. The currently implemented scenarios contain all communication functionalities and agent behaviours that are implemented. Besides some implementational issues that are caused by simple algorithms (see the next section), the system functions exactly as designed as far as simulation, communication and Multi-Agent behavior are concerned.

6.2 Analysis

As we have seen especially in the fire scenario, the currently used hypothetical location algorithm should be more refined to ensure that the Crisis Center sends an expert agent to the correct location. For the gas scenario it did work out correctly because the hypothetical location that is given by an exact average of all message locations gives a location that in this particular case is located downwind of the correct source, so that the gas cloud moves coincidentally over the hypothetical location so that when the DCMRExpert is sent to that location, by coincidence he reads the gas concentrations that prompt him to report an evacuation signal. This would not have happened if the hypothetical location was based on an average that was not exactly downwind of the source. Then the DCMRExpert would have totally missed the gas cloud and it would not have sent an evacuation signal. For the fire scenario it is more disturbing that the Fireman is being sent to a location where there is actually no fire. A possible suggestion for improvement for this could be that if a certain hypothesis is actually accepted, the hypothetical location from then on is only based on messages that support that hypothesis. In this way the reports from "odd" locations are filtered out and using an average function in this situation in this case will become more accurate.

Furthermore we also notice that in both scenarios the crisis center during the course of the simulation starts to send more and more messages, and every time instance the number of messsages becomes more and more. A lot of those messages are the same messages every time instance. The fact is that a lot of obsolete messages are sent to the agents in the field every crisis instance. This is because we implemented the decision rules in such a way that every rule in the rule base is being matched against all messages that came in since the start of the simulation. This is necessary for the forming of a good hypothesis because it needs all information that is available. For responding to messages however, it is not necessary to respond to a message that the Crisis Center has already responded to. Therefore the rules for generating responses to incoming messages should also be based on the time of sending of those messages. At this point this is not yet the case, but it is surely an improvement for efficiency. Then the Crisis Center only responds to the most recent messages.

Thirdly we can see that at this point the visualization only shows the messages that have come in so far. On a real Crisis Center map usually also some sort of hypothesis is being drawn. At this point the hypothesis is not yet visible on the map in MACSIM. Some sort of gas contour or visualization of a fire that dynamically is updated based on new incoming information is desirable in the next implementation iteration. A reference for graphical examples could be the SAFER Realtime system from SAFER Systems [SAF].

Finally, when the Crisis Center sends the message to the general public to evacuate, it currently just states that the general public should evacuate with the hypothetical location of the incident included. This results in the General Public running away in a random fashion. Some additional information for the general public why they should evacuate in the first place and to which direction preferably could be a welcome future addition. Therefore we can conclude that currently the first prototype of MACSIM is capable of simulating and interpreting simple crises. It seems to be a promising first step towards a

more advanced and elaborate Crisis Simulator in which more crises can be modeled. Before this can be achieved however, it remains necessary to test the current prototype more thoroughly with more and different kinds of scenarios, to improve existing and add new crisis modeling algorithms and finally to make the rules inside the Agents' rule bases smarter. Rules could for instance be made more elaborate and knowledge could be added based on more detailed information from domain experts. Also of course the features that due to time limitations are currently left out of the first prototype need to be added, to improve the usability of MACSIM.

Chapter 7

Conclusions and Recommendations

In this chapter we give a final roundup of the research conducted during this MSc. Thesis trajectory. In the first section we give a recapitulation of what has been delivered during this project. Then we try to give a more detailed answer to the questions posed in the first chapter of this thesis. We begin with answering the four sub questions, after that we give a final judgment of the project by formulating an answer to the formal problem definition. The evaluation effort is then being discussed and we try to formulate a trajectory for further evaluation. In the next section we give recommendations for improvements and further development of the system.

7.1 Conclusions

First we present answers to the questions as defined in the Problem definition in section 1.2. In the section after that we will focus more on the nature and recommendation for further evaluation of the current and future system.

For training in crisis situation, simulation is necessary. What should be simulated?

In the future, there will be an increasing demand for tool that can help in simulating crisis situations. MACSIM is able to simulate crisis situations based on predefined scripts that contain events. At the moment these events can be fires, explosions and gas escapes. Also the communication between agents is simulated. This is represented by messages that are sent and received by agents. This could model either regular conversations or mobile communication. The modeling of events takes place through the use of real time updating of calculations on formulae of algorithmic functions. The communication is modeled by messages using a useful crisis-based ontology in which the essential concepts can be expressed.

The current world model can be used for testing communicational structures, event updating algorithms and hypothesis forming. It remains to be said that more testing is still needed to discover to which extent the system will be able to achieve this. Also the crisis based ontology still need some additions to express crisis messages more realistically.

Can independent systems be integrated for simulating crisis management?

MACSIM is basically an integrated collection of systems, being a simulation world, an agent platform and respective agents, which also contain individual rule engines for reasoning. Therefore a lot of independent systems are integrated in the MACSIM design. Furthermore, the simulation component is designed in the form of a world consisting of many waypoints. These waypoints can be updated through independent update algorithms that are easy to adapt. It is possible to update existing algorithms or add new ones very easily. The current models are responsible for updating gas, fire and explosion events.

Theoretically any type of event could be added to this list of available events, provided that it is possible to make a time based updating model that can refresh physical changes in the environment in waypoints. If besides that it is assured that the waypoints have space reserved for the updating of that particular quantity, than virtually any model that represents a certain kind of crisis event can be added. This gives a lot of flexibility and allows a lot of creativity for designing scenarios. Using waypoints also reduces the computational complexity involved in an update instance.

On the other hand the waypoint based approach gives some problems with multiple events at certain location. Multiple events at one waypoint cannot be handled correctly in the current implementation. This is because the update algorithms are updating the waypoint independently from each other and they are at this point not designed to interact with each other, should a combination of events at a certain location normally result in another event.

Also we can see from the tests in the Evaluation chapter that agents cannot yet look into the distance. This is because of the fact that the world is based on waypoints; the agent can only observe things in its direct environment. He can for instance hear a bang, but not where it's coming from. This is mainly a lack of richness in the model. If the model is capable of providing more complete data to the agent about what is going on, i.e. more semantic richness is provided in the waypoints, the agents can observe those intelligent values and report more intelligently to the Crisis Center.

In the implementation chapter we noted that at this point the agents cannot yet update the world, because that is not yet implemented. This is of course an essential feature for future development of a crisis simulator. When this feature is added, more elements of crisis response can be evaluated.

How can we design a crisis simulator with a high sense of realism?

To ensure that we had data that was as realistic as possible, we held interviews with domain experts to receive as much information on crises in real life as we could. This meant reviewing real crisis training scenarios, properties of chemical substances that can be sensed based on chemical datasheets and gas dispersion models that were as realistic as possible. Also the communication structure was designed in such a way that it came close to the real situation. Because it is only a proof of concept, we could not implement every single aspect of crisis simulation as realistically as possible, but in future implementational iterations in other areas of simulation the same level of realism can be acheived, provided that contact is being maintained with the domain experts and intensive research is being conducted before during design and requirements analysis.

How to apply a Multi-Agent System for crisis management simulation? Are the properties of a Multi-Agent System suited for the communicational structure required for crisis management?

MACSIM is a JADE based Multi-Agent System that models to a certain extent the current communicational structure of crisis response. This means that the agents in this setup are not collaborative but more hierarchal in their design. The Agents in the field do their own things as long as the Crisis Center does not order them to do something else. The Crisis Center as an agent has a more commanding role than the agents in the field. However, the agents in the field provide at their own initiative the Crisis Center with information from the outside world that allow the Crisis Center to reason about a crisis hypothesis. So this system is not strictly a collaborative system, but it is achieving the goals of crisis response by working together, only in a hierarchical setting in which one agent is the commander. And because JADE agents are autonomous independent entities, using them inside a crisis response simulation provides similarities with people in the real world, and therefore agent representation can be a good model of a crisis response participant.

The communicational structure between the agents as it is implemented right now still leaves some room for improvement. Firstly the agents in the field do not yet communicate with each other, and the agents still have to know the names of the agents that they want to contact.

The ontology that is currently used for communicating also still needs some adaptations to make the messages more appropriate and realistic. This requires some changes in the design of the ontology but it means that some of the messages can now be broadcast correctly. Examples of this are the lack of dedicated slots in messages for medical problems or gas concentration in a gas, and also the inability to report something that is happening far away.

Can AI-reasoning help in simulating crisis management-related decision making?

Currently MACSIM makes use of rule-based observing in the field and rule base hypothesis forming inside the crisis center to determine what is going on and what is the hypothetical location. This is achieved through Jess rule engines with rules based on message input, with the right hand side consequences of those rules in Java code. This is done by means of JessListeners inside the JADE Behaviours. An advantage of this approach is the fact that consequences of rules and data operations can be run inside Java instead of inside Jess. This makes the consequences of firing rules much more easy to evaluate and to modify.

In the current version of MACSIM the Jess rules that supposed to represent a-priori knowledge of the participants in a crisis situation are still somewhat simple. With the additional information from domain experts that the agents are supposed to be modeling, it might be possible to devise more intelligent rules that closely resemble the real observation-based reasoning. Also some algorithms that are currently used inside the Java code for determining the hypothetical location and events can be made more sophisticated to ensure more accurate advice being sent from the Crisis Center to the agents in the field.

Problem definition: Is it possible to use a Multi-Agent System supported by AI-reasoning to develop a credible simulation of a Crisis and the corresponding crisis response by the authorities?

The final conclusion should be that the very first prototype of MACSIM, a Multi-Agent Crisis Simulator, is currently working and for the first phases of a crisis response can give a reasonable simulation. However, with still some components left out of the current prototype and some algorithms that still lack some sophistication there is still room for improvement but therefore there are also chances for significant performance improvements to allow this prototype to become a successful crisis simulator in the future. We believe that despite the primitive stage of the current prototype, MACSIM can part of the humble first steps towards realistic crisis response simulation, and that MACSIM demonstrates the first concepts.

7.2 Analysis Evaluation

The evaluation was done by implementing two scenarios with all the possible events included. These simulations were still quite short and modeled only part of the crisis response effort, because of the reasons mentioned in the Evaluation and Implementations chapter, but besides these details these scenarios worked reasonably well.

The main problem is that we were only able to test the current prototype with two scenarios, and to get a well founded judgment on the abilities, more tests have to be performed. But the results achieved with the currently performed tests still look promising.

Nevertheless it is important to evaluate in the future to which extent the use of scripts and scenarios as it is currently implemented can be used for the future requirements, such as the ability to dynamically add or modify events or the possibility to perform actions that modify the environment in which the crisis responders are functioning.

The most essential part of the evaluation will happen at the moment that the prototyping will reach the phase in which the users of the simulator will be able to fully interact with the system. Then intensive contact with the domain experts and future users is required to come to a simulation environment

in which the users, in this particular case the crisis responders in the field and the crisis managers in the crisis center, can experience a high level of presence and realism. It is therefore important that in the near future feedback is being received from the domain experts on the user interface suggestions presented in Appendix A.

7.3 Recommendations

7.3.1 Recommendations for Current System

The recommendations for a first prototype of a software product can usually often be characterized as: more is better. The more features in new development iterations are added, the more concepts can be introduced and the more the sense grows that a software product becomes complete. In the case of MACSIM more means: more models, more functionality, more agents and eventually more knowledge.

When we look at the problems of the previous section we can see that the problems can be divided into three categories: problems with features left out the first prototype, problems with existing features that are currently implemented but have to be improved and finally new features that appeared necessary during testing and evaluation. We will discuss all three categories in the following paragraphs.

At this point the agents in the field cannot update the waypoints yet. A solution to this has been described in the Design chapter, in the current implementation also space in the code is being reserved for this functionality, but at this point it is not yet implemented. The same holds for the lack of communication between the agents in the field. It has been designed it is only not yet implemented. The only solution to this is to implement it in the next design iteration.

Some current functionality still has to be improved. For instance the fact that agent names are hard coded in to the code is not a good thing. There is however an elegant solution to this problem provided by the JADE framework. It can be solved by using the services of the Directory Facilitator (DF). This service can provide a list of names of agents that perform a certain task to agents that want to send a message. Of course when an agent is added to the framework it should notify the DF that is provides a certain service, otherwise it still wouldn't work.

The hypothetical location determination algorithm is currently based on an average of the locations of all incoming messages. A suggestion to achieve a more accurate hypothetical location would be to in the future, when a hypothesis is being formed, only to take the locations of those events into account that are relevant. This relevancy can be topic based, or time based or event based. When you do the averaging (or another clustering algorithm) with a more appropriate set of locations, the location pinpointing should become more accurate.

The decision rules currently in use for observing the waypoint values and reacting to them are sometimes a little bit simple (Some examples of rules are shown in the appendix). With the intelligent combination of information and adding new a-priori knowledge to the rule engine, the overall behavior of the agents should be improving.

Finally there are some existing components that need to have additional functionality to improve their performance. For instance there needs to be more semantic richness inside waypoint model. If we want to describe an explosion, we can at this point only describe the fact that we hear a flash or a bang. We would like to have additional information like the direction where the flash or the bang is coming from. This kind of information would help the agent in the field with reasoning, but it would also help the Crisis Center in location algorithms for determining the hypothetical location. Of course this brings us to another problem: the messages sent by the agents should be able to send this improved information. This shows the significance of improving the messages currently sent by the agents, so more different concepts can be sent, and even already existing concepts that now currently do not have their own

dedicated slot can have their own slot.

The final problem that needs to be addressed is the problem that when waypoints are updated, the different events at the moment do not yet interfere with each other. This means that the effects of two different events can be happening at one location without having to do anything with each other. In some cases this might seem odd. The solution for this would be twofold. Firstly the events that are updating the waypoints should become more advanced so they can check whether certain other events are occurring and then adapt the updating accordingly. Secondly, when a certain combination of events should occur, it should become possible to add a new event to the simulation's script to indicate that a new unexpected event just happened. This concept is called dynamic scripting and is at this point still left out the design, but it could solve the problems that are currently experienced with multiple events.

7.3.2 Recommendations for the Future

If MACSIM contains more models, then it can simulate more events, and more different scenarios can be created. When more scenarios are created, this results in more possibilities to evaluate the system. More evaluation is still necessary to determine concepts or features that are still missing in the current prototype that can be added in new development iterations.

New types of agents will definitely have to be added. Examples of this could be a Police agent and Ambulance agent, or fir instance Medical Officer or a Mayor. These roles are very important in Crisis Management in the Netherlands, but at this point still left out of the design. With these new roles added in the system, more communicational schemes can be designed and new intelligence will have to be added to those agents. In all cases where new intelligence/ knowledge has to be added or intelligence/ knowledge should be updated, it is important to have discussions with the actual people that are to be modeled by the agent to get useful first hand experience and information that can be represented in rules, behaviors and agent actions. The rules in this way will also become better and more sophisticated.

When more features become available, there will inevitably also be a need to visually represent these new features. Therefore the existing user interface will need to be upgraded to improve the visual experience for the user that runs the simulation. At this point the user interface just shows message traffic, but real time crisis simulation of course is something that can provide a lot of opportunities for attractive visual representation.

In the distant future also mobile devices should be able to connect to the MACSIM platform. Theoretically, this should be possible because JADE is capable of connecting to for instance mobile devices. If that mobile device is capable of sending messages that are compliant to Siska Fitrianie's world model ontology, then other agents will not have any problems receiving and interpreting those messages. The only thing that needs to be tackled is the mobility of that particular agent and the connectivity to the JADE platform, which currently still is a single pc setup.

First priority, however, will have to be the addition of a more user friendly way to adapt and modify scenarios and simulation parameters. Also the storage of scenarios will be necessary, and was already part of the design, but not yet included in the current prototype.

Bibliography

[AFR] (Advanced Fuel Research) AFR. Advanced Fire Detection System Using Infrared Diagnostics. Web-Article.

[AHGG02] E.L. Anderson, E. Howlett, C. Galagan, and T. Giguere. CMSMAP: Oil, Chemical, Search and Rescue, and Marine Emergency Response Crisis Management System. *Proceedings of the 25th Arctic and Marine Oilspill (AMOP) Technical Seminar, Calgary.*, pages 1103–1114, 2002.

[Ald02] J. G. M. Alders. *Onderzoek Cafébrand Nieuwjaarsnacht 2001 (Report on the Fire in Volendam 2001)*. Ministry of Internal Affairs, 2002.

[Ale03] B. Ale. Inaugural Speech: Ons Overkomt dat niet, september 2003.

[ARI+04] Pyush Agrawal, Ingmar Rauschert, Keerati Inochanon, Levent Bolelli, Sven Fuhrmann, Isaac Brewer, Guoray Cai, Alan MacEachren, and Rajeev Sharma. Multimodal Interface Platform for Geographical Information Systems (GeoMIP) in Crisis Management. pages 339–340, 2004.

[BCTR05] F. Bellifemine, G. Caire, T. Trucco, and G. Rimassa. *JADE Programmer's Guide*, 2005.

[Bee04] M. Beelen. Personal Intelligent Travelling Assistant: a Distributed Approach. Master's thesis, Delft University of Technology, 2004.

[BL99] Tung Bui and Jintae Lee. An agent-based Framework for building Decision Support Systems. *Decision Support Systems*, Volume 25, Issue 3:Pages 225–237, 1999.

[BPR01] Fabio Bellifemine, Agostino Poggi, and Giovanni Rimassa. JADE: a FIPA2000 compliant Agent Development Environment. pages 216–217, 2001.

[Cai03] G. Caire. *JADE Tutorial for Beginners*, December 2003.

[CC04] G. Caire and D. Cabanillas. *JADE Tutorial: Application-defined content Languages and Ontologies*, 2004.

[COT05] COT. Op de grens van werkelijkheid - Observatierapportage oefening Bonfire (Observational Report on Bonfire Crisis Training), 2005.

[CPR92] CPR. *Methods for the Calculation of Physical Effects (CPR 14E) 2nd Edition*. Directorate General of Labour of the Ministry of Social Affairs and Employment, 1992.

[CPR97] CPR. *Methods for the Calculation of Physical Effects (CPR 14E) 3rd Edition*. Directorate General of Labour of the Ministry of Social Affairs and Employment, 1997.

[DD03] B. Dardelet and S. Darcy. Rescuing the Emergency - Multiple Expertise and IT in the Emergency Field. *Methods of Information in Medicine*, 4:6, 2003.

[DD06] U.S. DOJ and U.S. DHS. National Information Exchange Model (NIEM), 2006.

[DEC06] DECIS. CHIM Project Overview, 2006.

[DH05] H. Ducoutrieux and F. Haradji. Conception and Development of a Crisis Simulation System in the Dynamic Environment of a Simulated Car Traffic. Master's thesis, TU Delft, 2005.

[dJ68] W.J. de Jong. *Inleiding tot de Chemie*. Wolters-Noordhof, Groningen, 1968.

[Ebe99] U. Eberl. All Fired Up About Intelligent Detectors, 1999.

[EN04] EPA and NOAA. *ALOHA User's Manual*, 2004.

[ENotC] EPA, NOAA, and US Bureau of the Census. *Marplot 3.3 User's Guide*.

[EPA] EPA. CAMEO Chemical Identification Charts.

[FH03] Ernest Friedman-Hill. *Jess in Action*. Manning, 2003.

[FIP02] FIPA. Agent Communication Protocol Specifications. Technical report, FIPA (Foundation for Intelligent Physical Agents), 2002.

[FR06] S. Fitrianie and L.J.M. Rothkrantz. The Represenation of the World. Technical report, TU Delft, 2006.

[Geh03] H. Gehman. *Columbia Accident Investigation Board Final Report*. Columbia Accident Investigation Board, 2003.

[Gro04] R.J. Grootjans. A Simple Agent Framework for teaching AI Programming to Novices. Master's thesis, TU Delft, 2004.

[KBR05a] L. Kooijman, T. Benjamins, and L.J.M. Rothkrantz. Interview on Crisis Management with Leo Kooijman, MultiTeam Project Coordinator Rotterdam Fire Department. 2005.

[KBR05b] R. Kuijper, T. Benjamins, and L.J.M. Rothkrantz. Interview on Crisis Management with René Kuijper, Head Crisis Center DCMR. 2005.

[KHBV+04] T.H. Kean, L.H. Hamilton, R. Ben-Veniste, B. Kerrey, F.F. Fielding, J.F. Lehman, J.S. Gorelick, T.J. Roemer, S. Gorton, and J..R. Thompson. *The 9/11 Commission Report*. National Commission on Terrorist Attacks Upon the United States, 2004.

[Kla05] P. Klapwijk. TICS: A Topology based Infrastructure for Crisis Situations. Master's thesis, TU Delft, 2005.

[Kui06] René Kuijper. Real-life Complaints on gases incorporated in MACSIM, 2006.

[NBB02] NBBe. Module-Examen ROGS Hoofdbrandmeester Projectopdracht (Examination for Chemical Expert), 2002.

[NOA04] NOAA/EPA. Overview of ALOHA Algorithms, 2004.

[OdBE+01] M. Oosting, M.B.C. Beckers de Bruijn, M.E.E. Enthoven, J. de Ruiter, T.J.F. Savelkoul, and Y.I. Tümer. *De vuurwerkramp: Eindrapport (Final report on the Enschede Fireworks Disaster)*. Ministry of Internal Affairs, 2001.

[Rem05] J.W. Remkes. Evaluatie oefening Bonfire (Evaluation Bonfire training). *Report to the Dutch Parliament*, 2005.

[RN95] S.J. Russell and P. Norvig. *Artificial Intelligence, a Modern Approach*. Prentice Hall, 2nd edition, 1995.

[Rot06] L.J.M. Rothkrantz. Interactive Simulation in Disaster Management (ISDM). 2006.

[Rui00] M. Ruijten. *Interventiewaarden Gevaarlijke Stoffen (Intervention values for Dangerous Substances)*. Ministry of VROM, the Hague, 2000.

[SAF] (SAFER Systems Inc.) SAFER. *SAFER REALTIME/TRACE Technical Manual*.

[SBR06] J. Seen, T. Benjamins, and L.J.M. Rothkrantz. Interview on Crisis Management with Sjaak Seen, Deputy District Commander South Rotterdam Fire Department. 2006.

[Sch05] Paul Schooneman. ISME - Icon based System for Managing Emergencies. Master's thesis, TU Delft, 2005.

[SLOdS05] J. Seen, M. Lenssen, A. Overbeeke, and C. de Swart. GRIP op de Informatievoorziening (Information Sharing during Crisis Response). Master's thesis, NIBRA, 2005.

[SPD+05] N. Bellamine-Ben Saoud, B. Pavard, J. Dugdale, T. Ben Mena, and M. Ben Ahmed. An Agent-Based Testbed for Simulating Large Scale Accident Rescue Heuristics. *Journal of Computer Science*, (Special Issue):21–26, 2005.

[Tat06] I. Tatomir. 3MNews - Message-based Multimodal News. Technical report, TU Delft, 2006.

[TR04] B. Tatomir and L. Rothkrantz. Dynamic Traffic Routing using Ant Based Control. 2004.

[vdV04] J. van de Ven. COMBINED Crisis Management Scenario by DECIS. 2004.

[vHBR06] R. van Haagen, T. Benjamins, and L.J.M. Rothkrantz. Interview on crisis management with Richard van Haagen, Chemical Expert from the Rijnmond Environmental Agency DCMR. 2006.

[vV05] M. van Velden. ManetLoc - A location based approach to distributed world-knowledge in mobile ad-hoc networks. Master's thesis, TU Delft, 2005.

[Woo02] Michael Wooldridge. *An Introduction To Multi-Agent Systems*. Wiley, 2002.

[YD05] Y. Yuan and B. Detlor. Intelligent Mobile Crisis Response Systems. *Communications of the ACM*, 48:4, 2005.

List of Figures

List of Tables

Appendix A

Future Interface Design

MACSIM is a continuous work in progress, and the prototype that has been the result of this MSc project is the first step of towards development of a mature crisis simulation system.

For the initial design of MACSIM we had interviews with experts from the crisis management world. In these interviews we discussed what kind of information technology solutions people working in the crisis management business are actually waiting for. Based on these interviews we devised a set of concept interfaces that are ideas for future interfaces of new versions of MACSIM. These interfaces will also contain functionalities that are not yet implemented inside the current prototype, but in the future those functionalities will definitely be included.

Due to the fact that our version only was a prototype that only needed to be a proof of concept, we did not yet implement these interface ideas into a working product. Due to lack of time we also did not have time to receive feedback about these interfaces from the domain experts. Nevertheless the interfaces presented here will definitely become a source of inspiration during future development iterations. During those iterations there will be more than enough time to receive feedback about interfaces. The interfaces currently implemented in the prototype have a more diagnostic nature. Their main purpose is to demonstrate the current framework functionality of MACSIM. These interfaces are therefore only temporary.

There are two interfaces presented in this appendix: an interface for game participants in the field and one for game participants in a crisis center. Both have an entirely different function and outline. For the participants in the field the demands for a higher level of realism and the sense of presence are more important than for participants that want to train in a crisis center setting, which could be more a replacement of currently existing COPI training.

Not all proposed functionalities are shown in their entirety in these interfaces. This is because otherwise the screen would be too full. In cases in which the functionality is not shown it will be explained thoroughly in the text.

A.1 Interfaces for crisis responders in the field

A first idea for an interface for crisis responders in the field is shown in Figure A.1. It is a 3D-representation of the world as can be often seen in 3D-gaming applications as Quake or Unreal Tournament, but it is tailor made for crisis training purposes. The actors in the world are Agents, which can be controlled by the players, but also have a-priori knowledge of the world as they observe things in the world.

The user observes the world and the values in the waypoints automatically, but the initiation of actions is up to the user. When the user sees a fire for instance, the user can move his mouse pointer over it

and get the information about the fire that that user is able to see. This is of course depending on his observing skills. A fireman sees more information about a fire than an ordinary person. It therefore can perform more appropriate actions.

On the screen we see that part of the world that the user is currently looking at. In the top left corner we see a telephone icon, which allows the user to get into contact with other participants of the game. The participants can send messages to each other in which they discuss the status of the crisis, just the same as real crisis responders would do if they were using C2000 for instance. This area can also be used for showing incoming calls from other participants. The user can then decide whether he answers that particular call. In the top right corner a miniature map of the crisis area is shown, so the user has a sense of direction. Left to this is a compass that shows which direction the user is walking to. A controller is used to navigate through the world, usually in this type of environments this is a mouse. In the bottom of the screen the basic actions that can be performed are shown. The amount of actions an agent can perform varies per agent type.

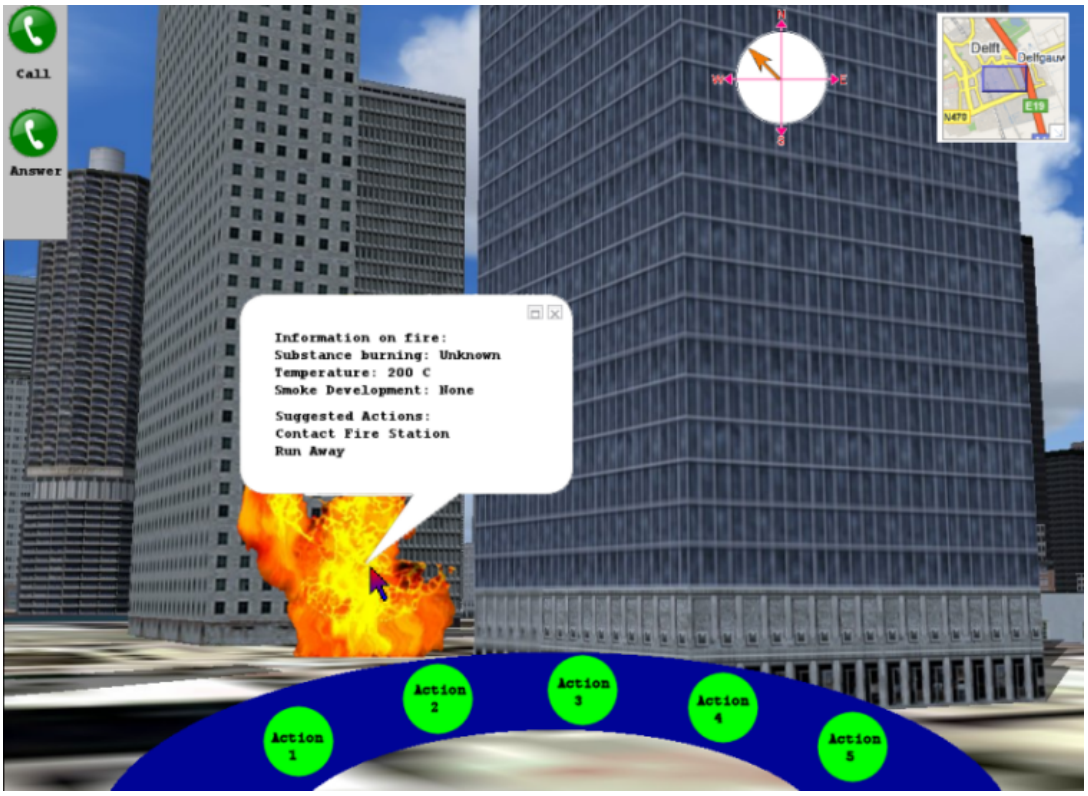


Figure A.1: Possible Interface for Agents in the Field for MACSIM

A.2 Interfaces for crisis managers in the crisis center

The users of the crisis center training simulation generally need a different kind of visualization. They need to see the global picture and therefore need to see what they usually see in a control room-type atmosphere. Only in this design, some extra features are added to simulate the information traffic and the more intelligent AI-functionality that we want to incorporate inside a future MACSIM interface. It is somewhat similar to the status map that is used in the DCMR environment to indicate what is going on (see Figure A.2).

A new functionality is the fact that you can directly contact units from the screen (trough the button in the top left corner). You can give the order to go to a certain place or ask agents in the field what the status is at a certain place. Also, like in the current prototype, the messages that come in from agents in the field or from the general public are placed on the map. The difference is that you can read the contents and other details of that message through clicking on the icon on the screen. Also the agents that are active in the crisis area are visible. If they say something, the icon lights up, and you can click on it. In a text balloon the content of the message in one way or another is presented. One can think of things like spoken messages, pictures or just plain text messages. The message then turns into an icon on the map, just like other messages.

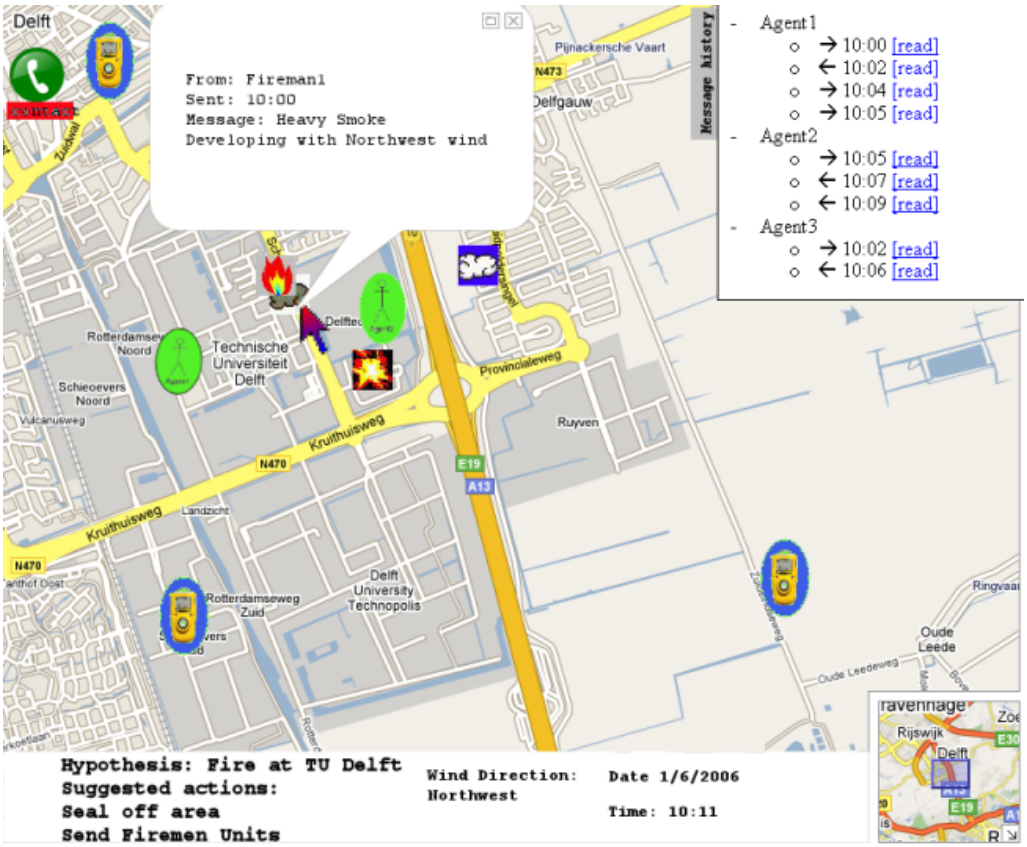


Figure A.2: Possible Crisis Center Interface for MACSIM

On the map also other important information can be shown. For instance, the location of chemical detectors can be added, which is also represented as an icon on the map. Just as with the regular agent icons, this icon lights up whenever a message is being sent. Also important information about buildings and other noteworthy places in the crisis area can be shown, just like it is the case in applications like Google Earth. This is practical for the people at the crisis center so they can combine information from multiple sources very easily. In the lower right corner of the map there is a mini map with the entire

area so the user can still see where exactly he is working.

For keeping track of the incoming messages, in the upper right corner there is also a message history that has in its storage all the messages. These are sorted by sender and time of receiving or sending. In this way a crisis manager can check messages for reference.

At the same time the AI functionality of MACSIM has to be shown. This is done through visualization. A contour or expected footprint of a gas cloud or the extent of a fire or a visualization of any other possible incident is shown on top of the map as far as the system can interpret it. Also the hypothesis, the date and time are shown at the bottom of the screen. Underneath the hypothesis suggested actions could be placed that are proposed by the system.

Appendix B

Interview Transcriptions Domain Experts (In Dutch)

Interview René Kuijper

Datum: 12 augustus 2005
Locatie DCMR, Schiedam
Aanwezig: R. Kuijper (DCMR) T. Benjamins (TU Delft), L Rothkrantz (TU Delft)

De heer Kuijper, voormalig voorlichtingsmedewerker en nu hoofd van de Hoofdmeldkamer (HMK), gaf een interessant exposé over het reilen en zeilen van de meldkamer. De meldkamer is 24 uur per dag te bereiken en heeft haar eigen robuuste verbindingen via telefoon (speciale lijnen), C2000, Interne mobilofoon).

Een van de hoofdtaken van de DCMR is registratie. Alle meldingen over incidenten die milieubedreigend zijn worden in eerste instantie gemeld bij de Hoofdmeldkamer. Het betreft hier meldingen van bedrijven maar ook (vooral telefonische) meldingen van burgers uit Rotterdam en omgeving. Kleine incidenten worden vaak via 112 gemeld bij de politie, bij grotere incidenten weten de burgers van de regio Rijnmond de DCMR snel te vinden.

De tweede hoofdtaak van de DCMR is verificatie, de meldkamer probeert in een zo kort mogelijk tijd een beeld van het incident te krijgen. Daartoe worden eigen inspectiediensten (de buitendienst) naar de vermoedelijke locatie gestuurd om polshoogte te nemen en eventueel aanvullende chemische testen uitgevoerd. Dit zijn wagens met meetapparatuur aan boord om aanvullende gegevens te verzamelen. Het betreft hier het nemen van lucht en watermonsters of van andere stoffen op de plaats van de ramp. In het geval van ernstige rampen mogen alleen brandweerlieden of medewerkers van de chemische dienst met beschermende kleding naar de plaats van het incident gestuurd. Om de verspreiding goed te kunnen vaststellen worden ook metingen verricht op specifieke gemarkeerde locaties. Op de meldkamer wordt op basis van modellen en beschikbare gegevens hypotheses geformuleerd en het bedreigde gebied in de vorm van een ellips gemarkeerd op een kaart.

De derde hoofdtaak is voorlichting van de bevolking. De Meldkamer informeert burgers die om informatie vragen tijden een klacht en doet dat waarheidsgetrouw op basis van de dan beschikbare gegevens. De meldkamer heeft ook een taak in het informeren van de bevolking. Aanvullend geeft de politie informatie met geluidswagens. De Meldkamer informeert tevens collega-diensten zoals Brandweer, Politie en andere betrokken zoals deelnemers aan het rampencrisisteam (RegOt) dat meestal bijeenkomt in het World Port Centre. (Gevestigd in de Rotterdamse haven bij Hotel New York.)

Op de meldkamer van de DCMR werkt men volgens het rampenplan wat is opgesteld in de regio Rijnmond. Gelukkig heeft men in de regio Rijnmond nog niet echt heel grote rampen meegemaakt waarbij alle plannen uit de kast moesten. Er wordt regelmatig geoefend, maar in de praktijk hebben ze nog nooit een situatie meegemaakt waar er grootschalige paniek uitbrak onder de bevolking. Hierdoor zijn alle situaties die er zijn geweest uiterst ordelijk verlopen en extreme crises zijn er dan ook niet geweest.

In geval van een crisis komen de politie, brandweer, de baas van de Meldkamer van de DCMR en de verantwoordelijke burgemeester samen voor de Co-Pi (Commando-organisatie Plaats Incident) vergadering. Hier wordt op topniveau in de regio alle diensten die verantwoordelijk zijn voor de openbare orde en veiligheid aangestuurd. Deze Co-Pi situatie wordt 18 keer per jaar geoefend en de Chemische dienst levert de basisinformatie voor de scenario's aan die in die trainingen gebruikt worden.

De heer Kuijper gaf aan dat het ergste wat er hier in de regio kan gebeuren is dat de chloortank bij AKZO chloor laat ontsnappen, hoewel dit uiterst onwaarschijnlijk is. In de 13 jaar dat hij bij de DCMR werkt zijn er 2 ernstige incidenten voorgevallen, te weten de brand op een schip in Vlaardingen in 2002 en het laten ontsnappen van een enorme hoeveelheid orthocresol uit een tank van VoPak in januari 2003. Dat stinkt ongelooflijk. (Het kwam er uiteindelijk op neer dat door een lasnaad die gebroken was er een klots orthocresol uit de tank gelopen was, die over de omringende beschermingswal sloeg. De bescher-

mingwal was wel berekend op een geleidelijk weglekken van een hele tank, maar niet op een plotseling weglopen van een deel van de tank.)

Officieel vindt dan het volgende plaats: eerst wordt de werknemer van een betrokken bedrijf geacht een incident aan zijn superieur te melden als hij denkt dat het incident schade kan hebben voor het milieu en de omgeving. In zo'n geval meldt het bedrijf een CIN-nummer door naar de DCMR/Politie. Chemicaliën hebben een code zodat iedereen precies weet om welke stof het gaat. Vervolgens vindt er een telefoongesprek plaats (soort telefonische vergadering) waar de Politie het woord voert, en waarbij ook de Brandweer, de DCMR, het Havenbedrijf en vertegenwoordigers van Verkeer en Waterstaat aanwezig zijn. De DCMR spreekt vanuit de meldkamer. De politie werkt in dit gesprek een standaardvragenlijst af met alle vragen die je maar over een incident kan stellen. Alle partijen wordt vervolgens gevraagd of er nog vragen zijn.

In de praktijk zijn de klachten van omwonenden in de regio Rijnmond altijd eerder bij de DCMR dan de meldingen van bedrijven. De omwonenden klagen meestal over stank, geluid, prikkeling van de ogen of van de slijmvliezen. De eerste klachten van bewoners komen meestal binnen 10 minuten na het incident binnen terwijl de meeste bedrijven daar minstens een kwartier voor nodig hebben om het via de officiële kanalen te melden. Dat komt vooral omdat het nummer van de DMCR bij 70% van de bevolking in het Rijnmondgebied bekend is. Als de eerste meldingen binnenkomen begint voor de binnendienst van de DCMR het interpreteren van de klachten. Hierbij komt vooral veel ervaring kijken. Deze klagen over stank, geluid, prikkeling van de ogen of van de slijmvliezen. [Hierbij zouden AI technieken echter eventueel wel een rol kunnen spelen.] Ze gaan hierbij uit van een startmal, die ze over de kaart heen leggen zodat de mensen van de DCMR die moeten gaan meten niet door mogelijk gevaarlijk gebied heen hoeven. Deze startmal geeft het risicogebied aan. [Dit is exact wat ALOHA op digitale wijze ook doet.]

Bij grote crises wordt naast het Co-Pi (het uitvoerend team) ook een RegOT (Regionaal Operationeel Team) geïnstalleerd die bevoegd is om te communiceren met de bevolking Daar zitten de hoofden van alle diensten in. Het oplossen van gevaarlijke situaties mag ook niet te rigoureus, want het Openbaar Ministerie wil indien nodig exact kunnen reconstrueren wat er plaats heeft gevonden en verklaart het gebied tot Plaats Incident. Dus de diensten mogen ook niet intensief aan het schoonmaken. Uiteindelijk bleek dat voor het VoPak-incident te betekenen dat de stank nog een half jaar merkbaar bleek in de omgeving van het complex. Ze mochten niet direct beginnen met opruimen maar wel een schuimdeken over de rotzooi spuiten.

In noodsituaties is de RET bevoegd om te evacueren. Hier wordt echter alleen toe besloten als het zin heeft, met andere woorden als er genoeg tijd voor is. Anders is binnenblijven met de ramen en deuren dicht het beste alternatief.

Om bewindslieden in het commandocentrum te krijgen is het soms handiger om ze per watertaxi of op de fiets te vervoeren vanwege verkeersdrukte. Bewindslieden krijgen geen zwaailicht. Medewerkers van de Chemische dienst van de DCMR hebben wel een zwaailicht. De chemische dienst is bevoegd om buitengewoon risicovolle metingen te verrichten. Deze afdeling van de DCMR staat dan ook onmiddellijk onder bevoegdheid van de brandweer in crisissituaties. Zij hebben ook beschermende kleding als het nodig is. De enige afdelingen die in geval van een terreuraanslag betrokken is, is ook de afdeling Chemisch Advies. De DCMR bemoeit zich namelijk in zijn geheel alleen maar met incidenten die milieueffecten hebben, dus aanslagen, hoe extreem ook, zijn voor de DCMR alleen van belang als er milieueffecten van zijn.

Meestal overlegt het RegOT in het World Port Center. Daar wordt ook overlegd of de sirene aan moet. De Brandweercommandant neemt deze beslissing. De politie verzorgt het contact met het publiek. Als de sirene gaat, dan moeten de mensen afstemmen op Radio Rijnmond. Deze rampenzender geeft officiële berichten door van het crisiscentrum. Maar de zender gaat ook onafhankelijk op zoek naar nieuws.

De zender heeft dus een dubbelrol. TV is nog steeds geen officieel medium voor rampenberichten. De DCMR raadt dus iedereen aan om een radio met batterijen bij de hand te hebben waarop je Radio Rijnmond af kan stemmen via de Ether.

Een groot probleem in de praktijk in het commandocentrum is het ontbreken van beeld. Tegenwoordig heb je op veel openbare plaatsen camera's hangen, maar niet overal hangen camera's, dus vaak moeten ze het zonder beelden stellen. De DCMR experimenteert tegenwoordig veel met webcamera's die ze over in de regio hebben opgehangen. Ze kunnen daarmee via Internet op plaatsen kijken waar normaliter geen camera's staan. De mensen kunnen tegenwoordig ook foto's opsturen via email. Dat kan de DCMR altijd helpen wat het visuele aspect betreft. In noodgevallen heeft men nog steeds een mobilofoon achter de hand voor als andere verbindingen weg mochten vallen.

Een klachtenplot is een digitale kaart waar alle klachten met stipjes worden aangegeven. Het rampengebied wordt via een ellips weergegeven, meldingen via bolletjes op de kaart, locaties waar metingen worden gedaan worden via bolletjes aangegeven. Windrichting wordt middels vlaggetjes weergegeven. Deze kaart wordt indien nodig uitgeruild met het commandocentrum. Feitelijk bestaat de periode na een incident uit 2 fasen: de eerste fase bestaat uit informatie van omwonenden en na een half uur pas zijn er meetgegevens binnen waardoor je iets preciezer kan zijn. De snuffelpalen in de regio geven vervolgens ieder uur een meting van de stoffen in de lucht. Vervolgens kan er nog gedetailleerd worden gemeten door de meetwagens van de DCMR ter plaatse of de brandweer. Daarvóór is de informatie van omwonenden het enige wat je serieus mag nemen. Mensen die klagen is de enige zekere informatie die je op zo'n moment hebt. Mensen weten wanneer iets stinkt of niet stinkt.

De moeilijkste klacht om vast te stellen is een boot die chemicaliën loost. De klachten komen steeds van een ander gebied in de regio, als er gemeten wordt lijkt de bron er niet te liggen. Pas relatief laat wordt de daadwerkelijke bron gevonden.

Het meest diverse aantal klachten tot nu toe was tijdens de stroomstoring van Shell deze zomer, waarbij het eigen noodaggregaat ook niet aanging. Dit heeft Shell miljoenen gekost en er moest toen extra worden afgefakkeld. Dit leidde tot klachten over stank. Het gaf geluidsoverlast, en er was sterke rookontwikkeling.

Iedereen in het rampenplan heeft eigen verantwoordelijkheid voor het uitvoeren van taken, tot het moment dat ze worden overruled door hun superieur. Ze moeten dan ook hun eigen veiligheid inschatten en als zij het idee hebben dat ze niet meer veilig zijn dan moeten ze het zekere voor het onzekere nemen.

Interview Leo Kooijman

Leo Kooijman werkt bij de Regionale Hulpverleningsdienst Rotterdam Rijnmond (RHRR) of wel de regionale brandweer van Rotterdam en omstreken. Tegenwoordig vallen de brandweerregio's (grotendeels) samen met de regio korpsen van de politie, uit praktische overwegingen. De taak van de regionale brandweer is uiteraard om hulp bij branden te verlenen. Dit betekent blussen en hulpverlening aan slachtoffers bieden, maar ook zijn ze bevoegd om de voorbereiding van rampenbestrijding te coördineren.

Tot augustus van dit jaar was hij hoofd crisisbeheersing. Hij moest alle betrokken diensten in crisis situaties met elkaar laten samenwerken. Die doen dat op het Plaats Incident (het CoPI). Juiste coördinatie binnen het CoPI is heel belangrijk. Hij weet dus wat er komt kijken bij crisismanagement op grote schaal. CoPI is één van de drie organisatiestructuren die bestaan in geval van een grootschalig incident. Er zijn in de rampenbestrijding 3 niveaus van opschaling, te weten:

CoPI (Commandostructuur Plaats Incident) - Alle operationele diensten zitten daarin. Zitten op het plaats incident, deze zijn bezig met de bestrijding van het incident zelf. Ze zitten in een in de buurt van het plaatsincident neergezette rode container.

RegOT (Regionaal Operationeel Team) - Ze werken meer op afstand. Dit is meer een aanvulling van het CoPI dan een opschaling. Het RegOT wordt ingesteld op het moment dat een incident heel lang gaat duren of van grote omvang is. Het doel van het RegOT is de gevolgen van het incident bestrijden op afstand. Ze denken vooruit in tijd en plaats. Dit betekent dat ze meer bezig zijn met de praktische zaken rondom rampenbestrijding. Zij houden zich bijvoorbeeld bezig met aflossing van de mensen op het CoPI en de hulpverleners ter plaatse of de planning van evacuatie.

Gemeentelijke Veiligheidsstaf - Hierin zit de burgemeester/lokale driehoek. In dit orgaan worden ook de moeilijke politieke beslissingen genomen. De Gemeentelijke Veiligheidsstaf is verantwoordelijk voor de communicatie naar buiten toe. Deze wordt alleen geïnstalleerd bij hele grote incidenten waarin een hele regio betrokken is.

De brandweer commandant is weliswaar operationeel verantwoordelijk, maar de Burgemeester is politiek eindverantwoordelijk, hij weet er soms niks van, maar hij moet toch de lastige beslissingen nemen die aan 2 kanten nadelen hebben. Dat kan ook prettig zijn omdat de brandweercommandant zich gewoon op het uitvoeren van zijn functie kan concentreren. In het World Port center in de Rotterdamse haven, waar Leo Kooijman werkt, komen het RegOT en de veiligheidsstaf van de gemeente Rotterdam bij elkaar.

Voorheen was Leo Kooijman ook verantwoordelijk voor veiligheidstraining bij CoPI's. Hij schreef dan een scenario wat kon gebeuren en dat werd dan nagespeeld. De procedures die ze bij de rampenbestrijdingsscenario's normaliter zijn vrij generiek, dat wil zeggen voor zoveel mogelijk gevallen toepasbaar. Ze doen dus zoveel mogelijk het zelfde, omdat dat tenminste nog trainbaar is en je dus tijdens een echte situatie niet hoeft na te denken. Het devies is: doe pas wat anders als het NUT kan hebben en alleen op een moment dat het uitvoeren van andere vastgelegde taken geen prioriteit heeft.

De trainingen voor de CoPI scenario's duren normaal gesproken 2 dagen. Dan doen ze 3 scenario's. Kooijman is dan degene die de gegevens aanlevert. Daar moet het CoPI wat aan het trainen is (meestal bij bedrijven) dan mee omgaan en aan de slag. Meestal bestaat er van een CoPI scenario voornamelijk een draaiboek van het berichtenverkeer. Het wordt dan een soort simulatiespel. De deelnemers krijgen eerst een voorgeschiedenis te zien op een film en daarna moeten ze met de inkomende gegevens aan de slag. Het is heel lastig om een training te organiseren omdat je, als je teveel vastlegt in een script je op een gegeven moment niet meer weet hoe je als scriptmaker (en regisseur) moet reageren als de deelnemers een andere denkrichting opgaan dan dat jij gepland had in je script. Ze maken tegenwo-

ordig minder rigide scenario's zodat de regisseurs ook nog kunnen ingrijpen als het de verkeerde kant op dreigt te gaan. Vroeger was dit wel een probleem.

Het is ook om deze reden dat integrale crisisoefeningen zoals Bonfire zo verschrikkelijk lastig te oefenen zijn. Dat was namelijk een oefening op verschillende rampenresponsieniveaus. Dan oefent dus een CoPI, een RegOT én een lokale veiligheidsstaf tegelijkertijd (in dit geval zelfs tot de minister aan toe) en het is al moeilijk om één zo'n orgaan te sturen, maar als deze organen vervolgens met elkaar in contact moeten komen tijdens hun oefening, dan moeten ze geen onverwachte dingen gaan doen anders is de organisatie van de oefening (onder meer Leo Kooijman) direct de regie kwijt. Het zou ideaal zijn als er dynamische rampenbestrijdingsplannen zouden komen, maar een hulpdienst moet te allen tijde weten wat hij moet doen, teveel tijd om te beslissen is er niet. Vaak is geen beslissing nog erger dan de verkeerde beslissing. Handelingssnelheid is dus geboden.

Het is voor rampenbestrijding bij de brandweer belangrijk om bij grote complexe incidenten, zoals rampen, de ramp in kleine overzichtelijke stukjes te knippen tijdens de planning. Die kleine stukjes zijn voor de mensen in het veld beter te overzien. Hierdoor kunnen zij doen waar ze goed in zijn en hoeven ze niet teveel tegelijkertijd na te denken.

Op het operationele centrum op het World Port Center is voor het RegOT GEEN TV geplaatst. (Of indien nodig staat hij op de achtergrond.) Daar hebben ze bij de rampenbestrijding alleen maar last van. Het kan zijn dat er een rookpluim te zien is en die wordt door de cameraploegen van een televisie station meteen vastgelegd, maar het gevaar van de situatie kan heel ergens anders liggen (de rookpluim zou noodzakelijk kunnen zijn en niet schadelijk), en om te voorkomen dat de focus van de rampenbestrijding teveel komt te liggen op die rookpluim die in beeld is, hebben ze liever geen tv-beelden in het RegOT. Het is daarnaast ook nog eens zo dat de televisieploegen tegenwoordig vaak eerder bij een plaats incident zijn dan de hulpverleningsdiensten, simpelweg omdat ze bij het eerste bericht van een ramp elkaar allemaal te snel af willen zijn vanwege de concurrentie.

Bij de Gemeentelijke Veiligheidsstaf, die de voorlichting naar buiten regelt, hebben ze juist weer wel de beschikking over een televisie, want zij kunnen dan de vragen van de pers beantwoorden over die rookpluim. Voor beide onderdelen van de rampenbestrijding bestaat er dus een eigen realiteit. Voor het RegOT kan die praktische realiteit iets anders zijn dan voor de Gemeentelijke Veiligheidsstaf. De rookpluim is voor de veiligheidsstaf dus wel degelijk van belang. Het is dus ook niet zo vreemd dat de Gemeentelijke Veiligheidsstaf en het RegOT in het World Port Center op twee gescheiden locaties zitten. Dat is nodig omdat anders die twee realiteiten door elkaar heen gaan lopen en dat is niet de bedoeling. Bij alle informatiebronnen die beschikbaar zijn voor een onderdeel van de rampenbestrijding moet altijd de vraag zijn voor wie de informatie nodig is om zijn taak goed uit te kunnen oefenen.

De huidige functie van Leo Kooijman (sinds augustus van dit jaar) is interregionaal Projectleider van het computersysteem Multi-team. Dit komt neer op het installeren van het systeem Multi-team, dat er voor is om de communicatie tussen verschillende diensten sneller te laten verlopen. Het systeem werkt via Microsoft Internet Explorer. Multi-team is meer voor het uitwisselen van gegevens via de computer. Het maakt géén gebruik van Internet, maar het is wel Internet Explorer-based. Het Internet is echter bewust niet als medium gekozen, omdat het altijd beschikbaar moet zijn en zelfs bij zoiets groots als het Internet kan dat nu eenmaal niet altijd gegarandeerd worden.

Op dit moment gebruiken ze bij Multi-team een speciale beveiligde lijn van KPN, maar ze gaan binnenkort overschakelen op OOV-net (OOV staat voor openbare orde en veiligheid). Dat is een netwerkverbinding die ook beveiligd is en waar alle it-diensten die door de overheid worden gebruikt voor veiligheidstaken binnenkort op over gaan schakelen. Bij Multi-team (dat geldt overigens ook bij alle andere diensten die op OOV-net gaan draaien) gaat het erom dat iedereen goed geautoriseerd is. Niet iedereen mag bepaalde files lezen. Alleen de mensen die bevoegd zijn om gegevens te bekijken mogen dat.

Door de steeds voortschrijdende technologie is het probleem van Leo Kooijman dat hij als projectleider van Multi-Team steeds het gevoel heeft dat er meer kan worden toegevoegd aan Multi-team. Hij vraagt zich continu af waar het systeem begint en waar het systeem zal eindigen. Wat nu state of the art (en duur) is, is een paar jaar later betaalbaar. Hij gaf het voorbeeld van een GIS (Geografisch Informatie Systeem,) wat een paar jaar geleden bij de ontwerpfase van Multi-team (1998) nog te duur was om te implementeren. Nu zijn GIS systemen zo goedkoop geworden dat ze makkelijk integraal onderdeel van een eventuele volgende versie van Multi-team kunnen zijn.

Er wordt op dit moment geëxperimenteerd met het plaatsen van een Data Terminal op een brandweerwagen. Hierop kunnen de eerste gegevens over een rampen situatie worden geplaatst, maar ook zaken als dynamische routebegeleiding, want een brandweerwagen moet binnen enkele minuten op de plaats van het incident aanwezig kunnen zijn. Met de files van tegenwoordig wordt dat soms lastig en daar kan dan via dynamische routebegeleiding op worden ingespeeld. Multi-team is ontwikkeld door TNO Defensie en Veiligheid, het is inmiddels al gebruikt in training en praktijk van CoPI's. Hierdoor zijn ze erachter gekomen dat er wel veel informatie voor handen is in crisissituaties, maar dat het wel slim moet worden gecombineerd. Ook informatie die onbelangrijk is voor de één kan van wezenlijk belang zijn voor anderen, maar hen bereikt die informatie nooit. Hij is nu bezig om voor de communicatie slimme scenario's te schrijven. Hierbij draait het vooral om het inschatten van hoeveelheden materieel, bv hoeveel bussen zijn er nodig voor evacuatie van een gebied.

Multi-team is een systeem dat naast C2000 bestaat. C2000 is het moderne alternatief voor de portofoon voor de "mannen in het veld". C2000 is geschikt voor communicatie via gespreksgroepen, je kan er 1 op 1 (P2P) gesprekken mee voeren maar ook 1 tegen mensen die in een groep zitten tegen praten. Het is dus een soort van GSM maar dan met nog extra functies. De eerste testen met C2000 in de brandweerorganisatie wijzen uit dat de organisatie van de communicatie erg goed moet worden geregeld. Je moet voorkomen dat er teveel gekakel op de lijn is, daarom is men nu aan het nadenken om de communicatieprotocollen aan te passen op het gebruik van C2000. De gesprekken moeten meer worden geordend.

Abstract Thesis Sjaak Seen

(During the interview, Mr. Seen told us about his Master's Thesis. We found that his thesis had more or less the same message as the interview he had just given, so we decided to include the Abstract of his thesis in this appendix instead of the transpript of the interview we had with him.)

From: Seen et. al., Grip op de informatievoorziening [SLOdS05]

Samenvatting

De huidige rampbestrijdings- en crisisbeheersingsorganisatie kent een drielagenstructuur (operationeel, tactisch en strategisch niveau) en voert terug tot de veronderstelling dat kritieke beslissingen op het hoogste niveau worden genomen. De organisatiestructuur is overgenomen uit de krijgsmacht en aangepast aan de andere 'vijand'. De structuur is echter ontwikkeld in een tijd waarin de communicatie via vaste telefoon- en faxverbindingen plaatsvond en computers, mobiele telefonie en moderne massamedia nog geen gemeengoed waren.

Maar door de stormachtige ontwikkelingen in de informatie- en communicatietechnologie is de klassieke drielagenstructuur onder grote druk komen staan: in rapporten en evaluaties van diverse incidenten wordt geconstateerd dat de communicatie niet de hierarchieke weg volgt en beslissingen op andere niveaus worden genomen. De media maken gebruik van uiterste geavanceerde communicatie- en beeldregistratietechnieken waardoor de officiële overheidsvoorlichting tijdens de acute fase van een incident achter de feiten aanholt. Bestuurders zien zich snel geconfronteerd met lastige vragen en zoeken naar alternatieve mogelijkheden om zo snel mogelijk de media-aandacht te kunnen bevredigen; en dat, terwijI de aan de kwaliteit van de verstrekte informatie gestelde eisen steeds hoger worden. Het onderzoeksverslag geeft een overzicht van de huidige invulling van de rampbestrijdings- en crisisbeheersingsorganisatie en aansluitend worden de innovaties op het gebied van de informatie- en communicatietechnologie en de maatschappelijke ontwikkelingen beschouwd. De beschouwing wordt ondersteund en onderbouwd door middel van casuIstiek en theoretische noties en per onderwerp afgesloten met een aantal subconclusies. In de analyse worden de subconclusies met elkaar gecombineerd hetgeen leidt tot enkele hoofdconclusies. Uiteraard besluiten we met enkele aanbevelingen ter verbetering van de kwaliteit van de organisatie van de rampenbestrijding en crisisbeheersing.

De hoofdthema's zijn:

- Besluitvorming op operationeel niveau
- Structuur van de rampbestrijdings- en crisisbeheersingsorganisatie
- De aansturing (command and control)
- Operationele informatievoorziening

Besluitvorming op operationeel niveau
Omdat de fysieke opschaling nooit gelijke tred kan houden met de urgentie tot handelen, moeten we accepteren dat in de acute fase van een incident strategische beslissingen worden genomen op het operationele niveau. Dit vergt echter een versterking van de professionaliteit van de operationeel leidinggevenden, maar ook de ontwikkeling van operationele vuistregels voor een verantwoorde mandatering van strategische beslissingen. Daarnaast moet de huidige moderne informatie- en communicatietechnologie worden ingezet om 'collegiale consultancy' met het strategisch niveau mogelijk te maken. Dit leidt tot een tempoversnelling in de beeld- en besluitvorming en bevordert een doortastend optreden.

Structuur van de rampbestrijdings- en crisisbeheersingsorganisatie
Door de mogelijkheden die de moderne ICT bieden, verloopt de informatievoorziening niet langer volgens de klassieke drielagenstructuur maar vindt er - met name in de acute fase en bij snel ontwikkelende, kortdurende incidenten - directe communicatie plaats tussen het operationele en het strategische niveau. De toegevoegde waarde van het tactische niveau komt nadrukkelijk naar voren bij langer lopende incidenten en rampen op het gebied van informatievoorziening, bijstand en aflossing, ondersteuning, etc. Nadere studie naar de inrichting van het tactische niveau als procesgerichte organisatie (gebaseerd op operatiën, informatie, planning, logistiek en ondersteuning) in plaats van disciplinegerichte organisatie is dan ook aan te bevelen.

De aansturing (command and control)
Ondanks de maatschappelijke tendens naar netwerkstructuren blijft een éénhoofdige leiding noodzakelijk voor de incidentenbestrijding. Echter niet vanuit een dominante discipline maar gebaseerd op procescoördinerende kerncompetenties. Dit betekent dus dat een medische ramp voor wat betreft de multidisciplinaire procescoordinatie niet onder leiding van de Regionaal Geneeskundig Functionaris hoeft te staan.

Operationele informatievoorziening
Eén van de belangrijkste pijlers voor een adequate en effectieve incidentbestrijding is de operationele informatievoorziening: de kwaliteit van de beeld- en besluitvorming wordt grotendeels bepaald door de kwaliteit van de beschikbare informatie. We constateren - in navolging van het ACIR-rapport - dat de operationele informatievoorziening slecht is georganiseerd, waardoor de media in de acute fase een monopoliepositie bezetten voor wat betreft de beeldvorming over de situatie en de effectiviteit van de incidentbestrijding bij bestuurders en burgers. Omdat de noodzaak tot transparantie is toegenomen (feiten kunnen niet langer worden achtergehouden, zelfs niet op de korte termijn) en de media in belangrijke mate bepalend zijn voor 'het rapportcijfer van de overheid' moet de operationele informatievoorziening op korte termijn sterk worden verbeterd.

Enerzijds dient de informatievoorziening te worden verbeterd door de toevoeging van een informatiemanager op de drie niveaus, waardoor de informatiehuishouding wordt versneld en geoptimaliseerd. Anderzijds dient de beschikbare informatie via de moderne informatie- en communicatietechnieken worden verspreid onder de betrokken partijen. Hierbij gaat het dus om een multidisciplinair informatiesysteem dat ook op afstand kan worden geraadpleegd, zodat alle betrokken diensten, organisaties en functionarissen gelijktijdig en 'in één oogopslag' worden geïnformeerd. Het informatiesysteem stelt de sleutelfunctionarissen in staat locatieonafhankelijk een eerste beoordeling van de toestand te maken alvorens zich ter plaatse of naar een actiecentrum te begeven. Hiervoor is een netwerkstructuur ("Web-based") nodig, waarbij de informatie zoveel mogelijk grafisch wordt weergegeven ("GIS-based").

Kritische succesfactoren om de operationele informatievoorziening te verbeteren zijn enerzijds eenheid, standaardisatie en aansluiting bij reguliere systemen én anderzijds voldoende kritische massa en tempo om de ontwikkelingen te kunnen realiseren. Het verbrokkelde staatsbestel zorgt er voor dat verantwoordelijkheden, bevoegdheden en financiële middelen over diverse partijen zijn verdeeld. Hierdoor wordt de organisatie en realisatie van een adequate, operationele informatievoorziening ernstig belemmerd. Eenheid en standaardisatie zijn noodzakelijk om informatiebestanden 'aan elkaar te kunnen knopen' en een actuele en adequate informatievoorziening te organiseren waardoor de benodigde gegevens op elk tijdstip snel en zonder overbodige ballast beschikbaar zijn. Zowel uit efficiency als effectiviteit dienen de informatiesystemen ook onder normale omstandigheden te worden gebruikt. Om deze ontwikkelingen met kracht en voortvarendheid ter hand te kunnen nemen, moet de informatievoorziening ten minste op het niveau van krachtige (veiligheids)regio's worden georganiseerd om de noodzakelijke investeringen te kunnen doen en de ontwikkeling voldoende professioneel te kunnen invullen.

Interview Richard van Haagen

Richard van Haagen is Medewerker Chemisch advies van de DCMR, in crisissituaties is hij chemisch adviseur ofwel ROGS (Regionaal Officier Gevaarlijke Stoffen). Hij moet dan advies geven aan Politie, brandweer, medische staf en bestuurders.

De taakverdeling is als volgt: in het geval van overlast, dan gaat de DCMR meten, als het gevaarijk is dan komt de brandweer in actie, in het grijze gebied tussen gevaar en overlast opereert de DCMR Expert. De DCMR heeft in een rampensituatie twee taken: officier gevaarlijke stoffen en Gasmeetplan leider. Het gasmeetplan komt in actie als er gevaar voor de volksgezondheid bestaat bij een incident. De gasmeetplanleider heeft bij uitvoering van het gasmeetplan de beschikking over DCMR mensen, meetwagens van het gemeentelijk havenbedrijf + meetmanschappen van de brandweer. Daar is de gasmeetplanleider de baas over. (70 meetploegen)

Van Haagen ziet de taak van adviseur gevaarlijke stoffen als een diplomaat tussen de politie en de brandweer en de geneeskundig officier. Bij botsende belangen is hij vaak bemiddelaar. Officieel heeft hij slechts een adviserende functie maar als de officier gevaarlijke stoffen iets zegt, moet je wel van goede huize komen om iets anders te doen. Het is achteraf bijzonder moeilijk om uit te leggen als bewindsvoerder dat je een advies van de chemisch expert hebt genegeerd. Wat heel belangrijk is voor een adviseur gevaarlijke stoffen is een netwerk van mensen en netwerk van kennis. Als je zelf iets niet weet, moet je iemand kennen die het wel weet en die ook snel kunnen bereiken.

Iets wat Van Haagen opvalt in het westerse denken is dat we als we maar alles in procedures vatten, zodat we dan zelf niet meer hoeven na te denken. Hij merkt tijdens crisissituaties dat als bestuurders hun papieren werkelijkheid maar onder controle hebben, dat ze dan denken dat ze weten waar ze het over hebben. Academici en Bestuurders verdienen meer, maar weten meestal niet waar ze het over hebben, terwijl mensen in de praktijk minder worden betaald en dat wel weten vanwege hun praktijkervaring. Mensen die alleen maar kijken naar procedures, cijfers en statistieken hebben geen hart voor de zaak. In de afdeling chemisch advies geldt: procedures zijn leidraden, geen dingen waar niet vanaf valt te wijken. Je moet wel blijven nadenken. Op het moment dat je van de regels moet afwijken moet je natuurlijk wel goed nadenken over de gevolgen, maar het moet in noodgevallen wel kunnen. Het moet geen non-optie zijn.

In het geval van een incident komen er meestal eerst berichten binnen van bedrijven die vantevoren waarschuwen dat ze iets gaan doen wat verkeerd zou kunnen zijn of dat er iets is misgegaan. In het geval van een ongeluk komen er meestal als eerste klachten van omwonenden binnen. Als reactie daarop gaan er daarna waarnemers kijken van DCMR. Vinden die dat er iets aan de hand is, dan rukt de DCMR met groot materieel uit. Er wordt dan ook een COPI ingericht, een Commando Organisatie Plaats Incident (in de regio rijnmond gebruiken ze het COPI, het is de bedoeling dat in Nederland alle regio's deze organisatievorm gaan gebruiken ter vervanging van het meer monodisciplinaire CTPI, Commando Team Plaats Indcident.) Een voordeel van COPI is dat een team beslissingen kan nemen, dit betekent dat iedereen verantwoordelijk is voor dat besluit, er wordt dan minder monodisciplinair gedacht. Er heerst dan minder het idee van "mijn straatje is schoon, mijn probleem is het niet." Een COPI zit in een vergadercontainer, een paar honderd meter van het plaats incident af.

Iets wat we in Nederland niet doen is een COPI om de originele COPI heen maken in het geval dat er een ramp gebeurt bij het COPI zelf, bijvoorbeeld in het geval van een aanslag op het COPI of besmetting. Dan zijn ze zelf ineens slachtoffer geworden. In landen als Spanje en Noord-Ierland (veel terreurervaring) hebben ze daar wel ervaring mee.

Er is een groot verschil in perceptie bij de bevolking over welke chemische stoffen gevaarlijk zijn. Waar bijvoorbeeld het meeste gedoe over is op chemisch gebied, zijn de LPG transporten om tankstations

te bevoorraden. De schade kan in het geval van een ramp wel groot zijn, maar de reden dat men dat gevaarlijk vindt schuilt voornamelijk in de regelmaat van de transporten. Volgens Van Haagen zijn de transporten van SO2 veel gevaarlijker, maar die zijn maar twee keer per jaar maar wel door de binnenstad heen in tankauto's. (SO2 wordt bijvoorbeeld gebruikt voor het wit maken van suiker.) Daar kunnen duizenden doden bij vallen in het geval dat zo'n tank ontploft en de wind een beetje verkeerd staat. De bevolking heeft eerder een idee dat een gas gevaarlijk is als gassen naar zwavel stinken. (Bijvoorbeeld alle mercaptanen.) Een veel gevaarlijker gas kan hierdoor door de bevolking niet opgemerkt worden, terwijl het er misschien wel is en minstens zo gevaarlijk kan zijn, alleen maar omdat het niet naar zwavel stinkt.

Richard van Haagen houdt ook examens en trainingen. Hij maakt examens voor het examen Officier Gevaarlijke Stoffen (ROGS). Zo'n examen vindt plaats onder verantwoordelijkheid van het Nederlands Instituut voor brandweerexamens, maar Van Haagen is verantwoordelijk voor de inhoud. Een ROGS examen bestaat uit een rollenspel waarbij men moet reageren op berichten die door allerlei hoofdpersonen kunnen worden gegeven. Hij mag ook het examen afnemen. Het examen toont aan wat echt belangrijk is. Communicatieve vaardigheden, werken onder druk, en in staat zijn onzin van echte feiten te scheiden zijn allemaal onderdelen. In dit examen worden veel berichten gestuurd van de mensen in het veld waarop de kandidaat dan moet reageren, die erg bruikbaar kunnen zijn voor een simulatieprogramma.

Hij is ook betrokken bij COPI traingen bij bedrijven: er zijn dan drie COPI containers tegelijk Dit is ook een simulatiespel waarbij bedrijven oefenen met de COPI-omgeving. Er komen in dit spel berichten binnen van onderaf (de mensen die je als COPI aanstuurt) maar ook de mensen die boven je zitten (bij voorbeeld de burgemeester of de directeur van het bedrijf) kunnen opbellen om je van je stuk te brengen. Het is in dit geval goed om tegen een directeur te zeggen dat je geen tijd voor hem hebt, terwijl hij bijvoorbeeld wil weten wat er aan de hand is. De meeste mensen voelen zich daarentegen vereerd dat ze de directeur aan de telefoon hebben.

Het COPI vergt een hoop training om het goed te kunnen doen, maar het heeft al voordeel dat als je als COPI werkt, je veel sterker tegenover juridische claims staat na afloop van een incident. Als een COPI iets besloten heeft, is het een besluit wat men met zijn allen neemt, het is een goed afgewogen besluit waar het beste alternatief is gekozen.

Hij is over het gebruik van ICT bij de rampenbestrijding niet al te positief. Hij pleit voor terughoudendheid. Hij vindt het makkelijk, maar het is gewoon nog niet betrouwbaar genoeg. Je moet ook nog gewoon boeken hebben als Chemisch adviseur om zaken in te kunnen opzoeken. Je kan gewoon niet altijd vertrouwen op electronica. C2000, het systeem wat gebruikt wordt voor de communicatie bij rampenbestrijding, is in principe handig maar de DCMR gebruikt ook nog de gewone mobiele telefoon terwijl van hogerhand wordt opgedragen om alleen C2000 te gebruiken.

Een nuttige ICT toepassing in de rampenbestrijding is volgens hem een systeem dat kan helpen een overvloed van informatie die binnenkomt te ordenen en de onzin eruit te filteren. Op alle verschillende bestrijdingsnivo's kan tijdens een ramp door dezelfde informatie een heel verschillend beeld ontstaan van wat er aan de hand is. Het zou goed zijn als een verschil in beeldvorming op de verschillende locaties in beeld gebracht zou kunnen worden. In de ene omgeving zijn ze in paniek, bij de ander niet (burgemeester tegenover bijvoorbeeld chemisch adviseur.) De burgemeester laat bijvoorbeeld de sirene gaan, maar de chemisch adviseurs snappen niet waarom, omdat er volgens hen geen reden tot paniek is.

Ten aanzien van een simulatie van een crisissituatie is hij sceptisch, mede vanwege het feit dat dingen in de praktijk bijna niet van tevoren te voorspellen zijn. Hij is bang dat een simulatie slechts het droog opvolgen van procedures zou worden, en dat is precies waar hij zo op tegen is. Als er in een simulatie de mogelijkheid wordt ingebouwd om buiten procedures om te kunnen werken,als de situatie daarom vraagt, zou dat een pluspunt zijn.

Appendix C

MACSIM Datastructure Examples

In this section some short examples are given how in the current code data structures can be edited. This is a quick reference, for more information on the functionality of MACSIM please refer to the Javadoc or source code.

C.1 Jess Rules examples

C.1.1 Agents in the field

In the rule bases of agents, rules are being used that are formulated as CLIPS rules, but then inside a Jess environment. Currently we use these rules in such a way that the left-hand side of a rule is used for pattern matching inside the rule base. This means that if there is a fact that applies to the filter imposed upon it, then the rule fires and the right-hand side of the rule is executed. In our case there is no right-hand side in the rule, because in our rule base the patterns are usually quite simple. This means that when an agent in the field sees something or receives a message, it is being put inside the Jess rule base as a fact, and when that fact contains properties that an agent is supposed to be looking for, a rule fires. This means that the Jess Listener that belongs to this agent notices that a rule has fired and that he has to execute the corresponding Java code. Here an example is given from a rule and the corresponding Jess Listener code from an agent in the field. New rules should be added in the form of a Jess rule with a name and then added corresponding Java code that should be executed in the JessListener.

Action rule(rules that lets the agent initiate an action)

```
(defrule retreat-Directive-1
    ;(Directive-issued)
    ?dir <- (Directive (order "RETREAT") (orderX ?x)(orderY ?y))
    (position (x ?posx)( y ?posy))

    (test (< (2d-distance ?x ?y ?posx ?posy) 50.0));kan ook nog andere waarde worden
    =>
    ;(do what Directive says)
    (retract ?dir)
    )
```

```
if (firedRule.contains("retreat-Directive-1"))
{
runAway();
if(messageContentStatus == IDLE)
   messageContentStatus = RUNNING_AWAY + "␣from␣" + (int) posX + "␣" +(int) posY+ "\n";
else
   messageContentStatus += RUNNING_AWAY + "␣from␣" + (int) posX + "␣" +(int) posY+ "\n";
}
```

Observation rule(The agent observes something and reports about it)

```
(defrule fire
    ?fire <-(fireLevel (fireLevel ?level&:(> ?level 0.01)))
    =>
    ; do whatever a fireman needs to do
    (retract ?fire)
    )
```

```
else if(firedRule.contains("fire"))
{
Report fireReport = new Report();
fireReport.setPriority(new Priority(1));
fireReport.setUrgencyLevel(new UrgencyLevel(1));
fireReport.setBeginTime(convertSimTimetoDateTime(currentDate, currentTime));
fireReport.setEndTime(convertSimTimetoDateTime(currentDate, currentTime));
PerspectiveArea pAreaFireReport = new PerspectiveArea();
pAreaFireReport.setCreationTime(convertSimTimetoDateTime(currentDate, currentTime));
pAreaFireReport.setLocation(new Location(new SpacePoint((int) posX, (int) posY), new
    SpacePoint((int) posX,(int)posY)));
Fire fire = new Fire();
fire.setLocation(new Location(new SpacePoint((int) posX, (int) posY), new SpacePoint((int)
    posX,(int) posY)));
fire.setBeginTime(convertSimTimetoDateTime(currentDate, currentTime));
fire.setEndTime(convertSimTimetoDateTime(currentDate, currentTime));
fire.setPriority(new Priority(1));
fire.setUrgencyLevel(new UrgencyLevel(1));
fire.setHazardousLevel(new HazardousLevel("evacuate"));
fire.setConfidence(1);
pAreaFireReport.addDynamicObjects(fire);
fireReport.setLocalWorldModel(pAreaFireReport);
reportOutbox.add(fireReport);
extinguishFire();
if(messageContentStatus == IDLE)
  messageContentStatus = REPORTING + "␣" +fire.getClass().getSimpleName()+ "␣" + fire.
      getHazardousLevel().getHazardousType()+ "\n";
else
  messageContentStatus += REPORTING + "␣" +fire.getClass().getSimpleName()+ "␣" + fire.
      getHazardousLevel().getHazardousType()+ "\n";
}
```

C.1.2 Crisis Center hypothesis rules

The crisis center uses incoming messages to determine what is going on in the outside world. It therefore needs to have rules that relate those incoming messages to potential events. Those rules are mentioned in the rule base of the crisis center. The rule base then starts pattern matching against the messages that are coming in. If a certain type of pattern applies, the rule fires and in the JessListener of the crisis center, scores that are in favour of a certain hypothesis are given.

```
(defrule fire-complaint
    ; eris een fire met location x,y
    ?fire <- (Fire  )
    =>
    ; storen in complaint,
    (store FIRE ?fire)
    (retract ?fire)
    )
```

```
else if (firedRule.contains("fire-complaint"))
{
    eventHypo.increaseEvent(FIRE,9);
}
```

C.2 Scenario

A Scenario is currently an instance of java.util.LinkedList containing a set of Event objects. The reason we use alinked list is that with this data structure it is very easy to add new event in between existing events later on (dynamic scripting). These Event objects have a start time, a location and a type. Those parameters are being set to give an EventGenerator the necessary information it needs to start an Event. Also that event needs a default duration. This is a RelativeTime object that indicates how long an event is supposed to last.

At this point a scenario is still hard coded inside the Simulation object. Later on this is supposed to be retrieved from a file or another form of storage. Then the getScript method should only return a Linked list, without having to add the events to them manually.

```
private Event event1 = new Event(new SimulationTime(14, 3), FLASH, 1000, 3400);
private Event event2 = new Event(new SimulationTime(14, 3), LOUD_BANG, 1000, 3400);
private Event event3 = new Event(new SimulationTime(14, 4), HEAVY_FIRE, 1000, 3400);
private Event event4 = new Event(new SimulationTime(14, 14), LOUD_BANG, 1000, 3400);

public LinkedList<Event> getScript()
{
    LinkedList<Event> result = new LinkedList<Event>();
    result.add(event1);
    result.add(event2);
    result.add(event3);
    result.add(event4);
    return result;
}
```

Waypoints and new values

The World class in the simulation world consists of a 2 dimensional array of Waypoints. These are storage objects that can contain all types of values. In a Waypoint values are stored that are updated because a model in the PhysicalModel class wants to update that particular value. It Asks to the waypoint getVALUE() to receive it or setVALUE() to set it, after calculating the new value.

The only thing that needs to be done to add a new value to the Waypoint is to add a new data type into the Waypoint class, define a getter and a setter. It is practical to define a default value for it as a constant in the class SimulationConstants. This is practical because it can be reached very easily from other classes that have statically imported this class, and to prevent NullpointerExceptions from occurring. When you have done so, you should add the following to the SetDefaultValues method in World: Of course you

```
newPoint.setValueName(DEF_VALUENAME_VALUE);
```

have to make sure that there actually exist events that updates that newly added value, otherwise it is of not much use.

C.3 Adding a new event

For adding a new event you have to determine many things:

- what is the name of the event (save it in SimulationConstants)
- what is the duration of the event (idem)
- what waypoint values should it update (defined in the Physical Model)
- in what fashion should it update the waypoints (via expansion of dispersion or in another way)
- which model should it use to update the waypoint values (add it into the physical model)
- what rules are there to recognize the event to use it as a hypothesis (add it to the rulebases and JessListeners of Agents)
- could it be a hypothesis (add it to the EventHypothesis class)

An event makes use of constants from the SimulationClass. It has a name stored there, an active time (duration) and, if it's a distinctive kind of event, also a hypothesis constant for the crisis center, so it can be named as one of the hypotheses after incoming events. Then it also needs to have rules that recognize it in the rule bases of agents in the field and the crisis center.

The best way to implement a new event is to track the code flow of the existing events to see to which extent the functionality of existing events can be applied to new events. It reaches from the method update() in the ScriptControl class to the PhysicalModel class where models should be placed.

How the agents react to this new event should be coded inside the JessListener and rule bases of the participating agents. In the case of the Crisis Center you should also add a new possible hypothesis to the EventHypothesis class. The algorithms to generate the hypothesis should be modified in such a way that it also incorporates the hypotheses of newly added events.

C.4 Making new agents

All agents in MACSIM are based on the same set of principles to observe and react. In a FSMBehaviour they sequentially execute a set of behaviours. First they get all the new messages and new data, then they interpret it and an action is executed. The core functionality can therefore be copied from other participating agents. Of course an agent must have a JessListener, and a set of rules to do something.

To receive waypoints, an agent controller should be added in the code of the Simulator class. This means that also a new value should be added in the Simulation class that determines the number of

that type of agent. This value is called numberOfAGENT_TYPE. When this is determined, you can then just copy paste the code from the addition of another type of agent controller which is mentioned above. It is important however that this new code is added ABOVE the comment block in the simulator code. This is mainly to enable new agents to be added and not to interfere with the controller of the CrisisCenterAgent, which should always be added last. An interface that might be needed can always be added.For communicational behaviour it is recommended to look at the other Agents and copy and paste what is needed.

C.5 Simulation constants

The used constants are included here to show what kind of constants can be shared in the MACSIM system. This can give room for suggestions of new types of values or constants that need to be shared.

```
// map constants
// TODO echte mapsize
public static double MAPSIZEX = 5000;
public static double MAPSIZEY = 4000;
public static double MAPGRID = 20;
// Event constants
public static final int GAS_DISPERSION = 0;
public static final int LOUD_BANG = 1;
public static final int FLASH = 2;
public static final int SMALL_FIRE = 3;
public static final int HEAVY_FIRE = 4;
// Gas type constants
public static final int CYANIDE = 0;
public static final int HCL = 1;
public static final int METHYLMERCAPTAN = 2;
public static final int NITROGENTETROXIDE = 3;
public static final int TRICHLOROSILANE =4;
public static final int TOLUENE =5;
public static final int ETHYLENE =6;
public static final int BUTYLMERCAPTAN =7;
public static final int SULFURDIOXIDE = 8;
public static final int OZONE = 9;

// Measurement unit constants
public static final String PPM = "PPM";
```

```
// Observingskills constants types of observations that are
// possible inside the world
/*
 * What do these quantities mean?
 * TEMPERATURE = temperature in degrees C
 * WIND_SPEED = wind speed in m/s
 * VISIBILITY =
 * BANG = hear a bang or not
 * FLASH_LIGHT = see a flashlight or not
 * GAS_CONCENTRATION = gas concentration in Kg/M^3
 * GAS_COLOR = the color of a certain gas that is spreading
 * GAS_BEHAVIOUR = room for special details of the gas (UNDEFINED YET)
 * GAS_VISIBILITY = true or false;
 * GAS_SMELL = different types of smell can be defined for each int
 * GAS_DANGER =
 * GAS_TYPE =
 * SPECIAL_OBJECTS = Each int can represent special objects (UNDEFINED YET);
 * FIRE_LEVEL = a double between 0 and 1 representing the level of fire;
 * SMOKE_LEVEL = (UNDEFINED YET);
 * WIND_DIRECTION =
 * GAS_VECTOR =
 */
public static final int TEMPERATURE = 0;
public static final int WIND_SPEED = 1;
public static final int VISIBILITY = 2;
public static final int BANG = 3;
public static final int FLASH_LIGHT = 4;
public static final int GAS_CONCENTRATION = 5;
public static final int GAS_COLOR = 6;
public static final int GAS_BEHAVIOUR = 7;
public static final int GAS_VISIBILITY = 8;
public static final int GAS_SMELL = 9;
public static final int GAS_DANGER = 10;
public static final int GAS_TYPE = 11;
public static final int SPECIAL_OBJECTS = 12;
public static final int FIRE_LEVEL = 13;
public static final int SMOKE_LEVEL = 14;
public static final int WIND_DIRECTION = 15;
public static final int GAS_VECTOR = 16;
```

```
// Waypoint default start constants
/*
 * These constants are the default starting constants
 * for all Waypoints when the simulation starts.
 *
 * if the values haven't changed by the user in any way,
 * the world loads these values in
 * every Waypoint while creating the waypointArray.
 */

public static final double DEF_TEMPERATURE = 20;
public static final double DEF_WINDSPEED = 1;
public static final int DEF_VISIBILITY = 2;
public static final boolean DEF_BANG = false;
public static final boolean DEF_FLASH_LIGHT = false;
public static final double DEF_GAS_CONCENTRATION = 0;
public static final int DEF_GAS_COLOR = 0;
public static final int DEF_GAS_BEHAVIOUR = 0;
public static final boolean DEF_GAS_VISIBILITY = false;
public static final int DEF_GAS_SMELL = 0;
public static final boolean DEF_GAS_DANGER = false;
public static final int DEF_GAS_TYPE = 0;
public static final int DEF_SPECIAL_OBJECTS = 0;
public static final double DEF_FIRE_LEVEL = 0;
public static final int DEF_SMOKE_LEVEL = 0;
public static final double DEF_WIND_DIRECTION = 315.0;

// Default lengths of events.

/*
 * Here the default length (in Relative Time format) of an event is noted for all the events
     that are possible.
 * Dispersion events only have an Active time, that is the time that the world needs to
 * be updated. Expansion events have a Threshold Time, e.g. how long it takes to expand the
 * event to the adjacent Waypoints.
 *
 */

public static final RelativeTime GAS_DISPERSION_ACTIVE_TIME = new RelativeTime(0, 10, 0);
public static final RelativeTime FLASH_ACTIVE_TIME = new RelativeTime(0, 0, 2);
public static final RelativeTime LOUD_BANG_ACTIVE_TIME = new RelativeTime(0, 0, 2);
public static final RelativeTime SMALL_FIRE_ACTIVE_TIME = new RelativeTime(0, 3, 0);
public static final RelativeTime HEAVY_FIRE_ACTIVE_TIME = new RelativeTime(0, 3, 0);
public static final RelativeTime SMALL_FIRE_THRESHOLD1 = new RelativeTime(0, 0, 5);
// new RelativeTime(0,1,0);
public static final RelativeTime HEAVY_FIRE_THRESHOLD1 = new RelativeTime(0, 0, 5);
// new RelativeTime(0,0,20);
public static final RelativeTime HEAVY_FIRE_THRESHOLD2 = new RelativeTime(0, 0, 7);
// RelativeTime(0,0,40);
```

```
// Smell constants
public static final int BITTER_ALMONDS = 0;
public static final int SHARP_ODOR = 1;
public static final int ROTTEN_EGGS = 2;
public static final int AROMATIC_ODOR =3;
public static final int SWEET_ODOR =4;

// ontology colors
public static final Colour ONTOCOLORLESS = new Colour(255,255,255);
public static final Colour ONTOBLUE = new Colour(204,255,255);
public static final Colour ONTOBROWN = new Colour(102,51,0);

// Hypothesis constants
public static final int EXPLOSION = 0;
public static final int GAS_ESCAPE = 1;
public static final int FIRE = 2;

// Color Constants
public static final int COLORLESS = 0;
public static final int BROWN = 1;
public static final int BLUE = 2;

// Directive Constants
public static final String GOTO = "GOTO";
public static final String RETREAT = "RETREAT";

// Agent Status Constants
public static final String RANDOM_WALK = "random␣walk";
public static final String RUNNING_AWAY = "running␣away";
public static final String ON_THE_WAY = "on␣the␣way";
public static final String IDLE = "idle";
public static final String COMPLAINING = "complaining";
public static final String REPORTING = "reporting";
public static final String WALKING_TO = "walking␣to";
public static final String DRIVING_TO = "driving␣to";
```

Appendix D

MACSIM Paper for CGAMES'06

MACSIM: Serious Gaming in Crisis Management via Script-based Simulation

T. Benjamins, drs. Dr. L.J.M. Rothkrantz,
Man-Machine Interaction Group
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology,
The Netherlands

***Abstract*— Experiments in crisis management are expensive and difficult to realize. There is also a lack of training facilities in real crisis environments. Serious games and simulation can provide an alternative. We developed a system that provides serious game for crisis management. It is called MACSIM (Multi Agent Crisis Simulator Interpreter and Monitor). It is composed of the following components: First a software based platform for dynamic simulating of disasters. We designed a communication infrastructure that allows agents participants in the simulation to exchange messages. Every agent is able to observe the results of crisis events, process these events using dynamic scripting (based on XML files) and initiate appropriate actions. The decision making process is distributed among autonomous agents. We developed a first prototype. The design and test results will be described in this paper.**

Keywords: gaming, serious gaming, simulation, crisis management, multi agent systems

I. INTRODUCTION

IN recent years we observe an enormous growth in the scale and complexity of the terrorist attacks and natural disasters. It proves that the existing infrastructure and crisis management is unable to face the challenges of such events. There is a need for additional research, training methods and training facilities. But to set up a test in real life crisis situation is far from trivial. To research for example communication and corresponding infrastructure during terrorist attacks or flooding is infeasible. Software from the game industry can be applied to crisis context, to provide a virtual simulation environment for research or an interactive method for training in the field of emergency response.

In general games are designed for entertainment. Our application domain is more serious. To play a game on terrorist attacks can be very exciting and joyful for the players. But if rescue employees as fireman and police and first aid employees use the game for training, the ultimate goal is to have better trained people to help more victims. So the games will be employed for serious goals.

For many years disasters are simulated for training and research. Disasters are generated using a fixed script, based on an XML file. In serious gaming dynamics scripts are used. Serious games are interactive. Users can change the order of the crisis events and the environment has to be adapted taking care of the reactions of the user.

Games are usually based on a phantasm script and generate a phantasm world. Serious games are supposed to have a high reality and presence value. Players should experience the environment as a real world and should get the feeing that they are part of this world. So in serious gaming the environment is not full of dragons and ghosts. And the actions are modeled after human behavior. In the current serious game it is possible to simulate an explosion of a chemical plant. The toxic cloud spread out according to the law of physics. And in case of a fireman flushing a fire should result in a decrease of the fire.

Recently the COMBINED (Chaotic Open world Multi-agent Based Intelligently NEtworked Decision-support Systems) project [10] was finished, which was an initiative of DECIS-labs, in combination with several Dutch research partners, including TU Delft, The university of Amsterdam, TNO, Thales and several Fire departments (NIFV), . The goal of this project was to design systems that consist of human actors and artificial agents. These agents work together to achieve their common goals, even if this means they are functioning in chaotic circumstances. Examples of other work in this field can be found in [11] and [12].

Our research conducted for MACSIM fits in the framework of the COMBINED project and had three main goals: to create a crisis environment in which different crises can be simulated by means of an event generator, to create a communication layer through which agents can exchange messages, and to create intelligent software agents that move around in the crisis world. This crisis world should be the basis simulation and training based on serious gaming.

At this moment game software is available which can be used for serious gaming. This software provides tools to build a realistic environment and to design bots with realistic behavior. We decided to develop our own software. First software tools are not freely available. Games which are able to generate realistic crisis events are not well developed. Next

our focus was how to design the communication between agents and to model agents. In future implementation iterations gaming software will be used to design 3D-graphic visualization to represent the data currently available in side the simulation world.

This paper will have the following structure: In section II we will introduce the world model for MACSIM. The related literature and sources of research will be discussed in Section III. Section IV will be dedicated to introducing the designed software components and the way external software components like JADE and Jess was included. In the Evaluation section (Section V) an example of a testing scenario is given, and finally we end this paper with a conclusion.

II. Related work

At this moment we observe an explosion of serious games initiatives, tools and games. Many of them are developed in the military domain. "America's Army" and Darwars Ambush are on of the most popular serious games. Another example is "Incident Commander" which teaches incident management for multiple scenarios's, including terrorist attacks, and natural disasters. Both games are built on the top of other games, using the game engine. America's Army is built on top of Unreal engine and Darwars Ambush on top of Operation Flash Point. In this section we will further restrict ourselves to the simulation and games which are on the basis of our research.

The US government organizations EPA (Environmental Protection Agency) and NOAA (National Oceanic and Atmospheric Administration) have made a very easy-to-use gas dispersion modeling tool called ALOHA (Arial Locations Of Hazardous Atmospheres) [6] . It is able to model chemical releases and has some advanced features in modeling. ALOHA has been designed for people with crisis response duties, so after an incident they can get a fast overview of what is going on and what will be the situation in the upcoming hour.

A program that is especially dedicated to real time simulation is the program REALTIME, which is made by Safer Systems [7]. Once a branch of DuPont de Nemours, one of the leading chemical companies in the world, it is now an independent company that is dedicated to chemical risk estimation and chemical simulation. REALTIME is a program that is used by a lot of chemical companies for determining the consequences of an incident that just happened. It can also be used as a simulator.

Hazmat: Hotzone is a simulation that uses video game visualization techniques to train first responders about how to respond to emergencies concerning hazardous materials. It is currently in the development stage at the Entertainment Technology Center at Carnegie Mellon University in collaboration with the Fire Department of New York. Hazmat: Hotzone is still in development and is estimated to be completed in Spring 2007 [8].

Another project focused on crisis situations is the Rescue project. We agree with the researchers on the following statements as can be found on the project website [9]: The response to different types of crises in a timely and effective manner can reduce deaths and injuries. It is also important to contain or prevent secondary disasters, and furthermore try to minimize the resulting economic losses and social disruption. The drillsim simulator, which is a part of the CAMAS (Crisis assessment, mitigation and analysis)-testbed [13] is a multi-agent crisis simulator in which crisis response roles (e.g., evacuation) can be played by game participants. The simulator models different response activities at both tactical and operational levels, and model the information flow between different crisis response agencies. External components can be inserted at different interfaces between these activities or at some point of the information flow in order to study the effectiveness of research solutions in disaster management and tested for utility in disaster response. In addition this simulator can integrate real life drills into a digital simulation.

III. Model

To describe the concepts in MACSIM, we have devised a World Model that is divided into three parts (see Figure 1), being the Real World, the Observed World and the Crisis Center. Through these parts of the world model we will explain the MACSIM concepts.

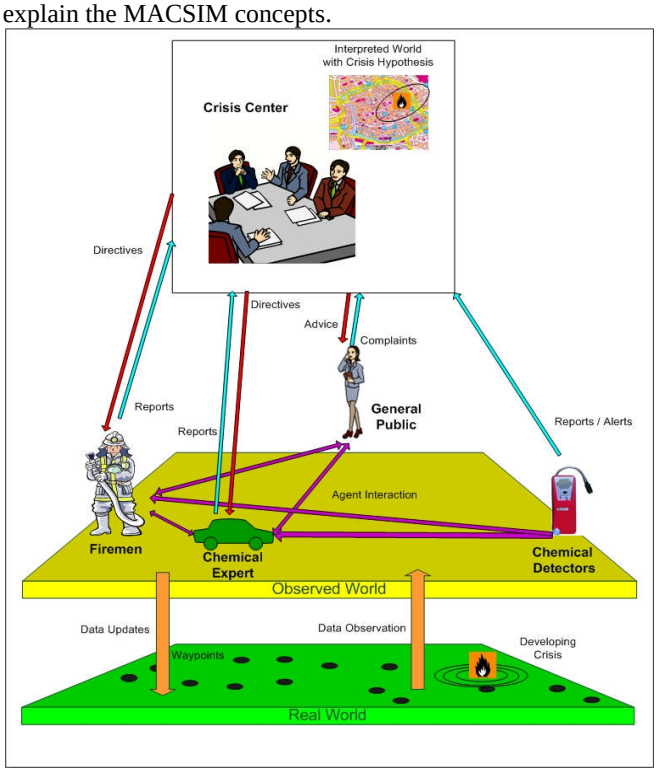


Figure 1: Global Overview of MACSIM

Real World

The first world is called the Real World. We consider the Real World to be a simulation of the real world, that is, a model of the real world good enough to fit our needs. In this world there are a lot of waypoints. Waypoints are considered to be data storage with information about the area directly surrounding a certain point. This data can also be updated. These waypoints contain a lot of data that the agents walking through the area use to gather information about the crisis at hand. Those waypoints are located all through the area. The main reason we use waypoints is because of the reduction of computational complexity that is achieved when we only have to calculate physical properties for a limited number of points in the world instead of for every possible coordinate in the entire area. As long as the computations for a certain point are reasonably accurate it will serve our purpose very well. While the simulation lasts, the physical properties of a waypoint are subject to change.

Observed World

The second world is called the Observed world, because it is in this world all the agents see the events unfold and report about it to the Crisis Center or to other agents. In the Figure we can see that the different agents can report things to each other. Also they can complain or report observed effects of events to the crisis center.

In the Observed World operate all different types of agents that have one thing in common: They observe the world. But because they are all have a different perspective on the world, they interpret the effect of certain event differently, and therefore also react differently to those effects. This leaves room for flexible interpretation of situations that in real life is one of the key characteristics of crisis response.

In the observed world operate different types of agents that all have their own view of the situation. This is visualized more clearly in Table 1.

Table 1 Agents and corresponding Actions

	Sensors	**Observers**	**Professionals**	**Decision Makers**
Message passing	Reporting	Reporting Receiving	Reporting Receiving	Ordering Reporting
Reasoning	N/A	Common Knowledge	Analysis based on experience	Analysis Incoming Information
Tactical Actions	N/A	N/A	Orders to peers	N/A
Operational Actions	N/A	N/A	Job-Related Response	Orders to Crisis responders

The table contains the following agents:

- ***Sensors***: Devices that are placed inside the world to report to other agents what is going on over there. This could be gas detectors, or smoke detectors or more intelligent devices reporting through intelligent combinations of sensory input.
- ***Observers***: The innocent bystanders that mind their own business but when something happens in the environment get involved in the crisis. They can be victims, but also complain about certain physical properties that cause them to have medical problems.
- ***Professionals***: These agents have to perform a job during a crisis situation and all their actions are based on performing that job as good as possible. Furthermore they maintain contact with the crisis center and give aid and information about what is going on to innocent bystanders. Examples of this kind of agents are Firemen and Chemical Experts.
- ***Decision Makers***: Decision Makers have the intention to control and to be in charge of certain professionals in the field. They are the ones politically responsible for the crisis response effort and therefore it is absolutely vital that they make good decisions. Based on either their own observations (a commander in the field) they form an interpretation of the world in which their decision will have to meet their responsibilities.

Crisis Center

All information acquired in the Observed World is being sent to the Crisis Center. The Crisis Center will be able to either inform the agents in the world about what's going on or it will send directives. The crisis center has to determine what is going on in the real world based on the views that the agents in the observed world have of it. It has to make a reconstruction of the facts given by the observing agents. Based on this a hypothesis of the crisis situation is formulated.

If they have a theory about the crisis, the Crisis Center can decide whether or not action should be taken and if so, what kind of action. This means that they can request Professionals to check out a current location about what is going on. If the investigations of those Professionals give enough evidence that there is a problem, they could order further units to go to the presumed location and aid in the crisis response effort. Also advice to the general public is being given about the situation.

Communication

The data flow between different components of MACSIM can be organized in a view as shown in Figure 2. In this view we can distinguish a Simulation Layer, Agent Middleware, an agent layer and one or more GUI's. It gives a clear overview

of the flow of information and in which way it is being transferred to the components in the system.

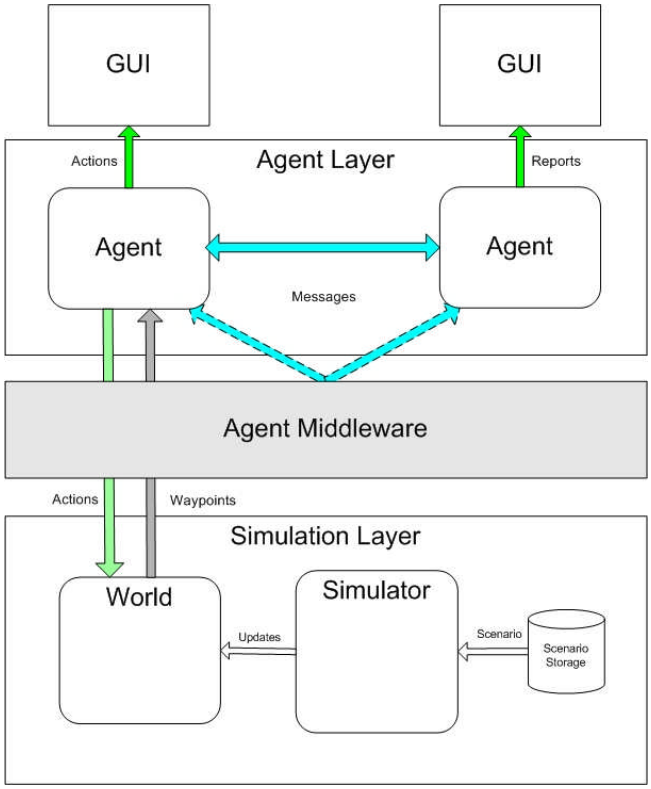


Figure 2: Global Overview Communication

The Simulation Layer contains scenario storage. In this storage scenarios are being kept. A scenario can be acquired from the storage by the Simulator. The Simulator is in charge of simulating crises and for that it needs scenarios. Those scenarios are being transformed into a script internally by the simulator. A script is basically a timeline with a start time and an end time and certain events that can take place in the world in between. The simulator is processing those events and this usually means that as a result of a certain event the world is modified in one way or another, i.e. the simulator is updating.

When the world is being updated, the agents in the world should be notified, because they have to sense changes in the world caused by the events. This is where the agent middleware comes into play. The agent middleware takes care that the agents in the simulation are receiving the updates of the world. The agents are receiving this information via agent middleware of because they are supposed to be autonomous. This means that the agents are supposed to work as independently as possible. Therefore other components should not have direct access to the agents, because that would imply some sort of ownership that does not fit inside the concept of independent agents.

In the meantime, the agents are receiving data updates of the world in the form of waypoints. If these agents sense this data they can process and reason about it. Based on this reasoning the agents initiate actions. Those actions might have an effect on the world or not, but this is of course depending on the type of action that is the result of the agent's reasoning.

Besides reading waypoint data, the agents are also capable of sending messages to other agents. In the diagram the blue arrow indicates this, but it would be more accurate to connect the arrows via the agent middleware. This is because of the same reasons of agent independency. Those messages are being sent to other agents through the agent middleware as well. The agent actions that have an effect on the world are being propagated back again to the world to implement the changes. This requires a synchronicity scheme that ensures that the simulator applying the script to the world knows about the updates by agents, so it can update the world again according to the most recent changes.

Finally the users of MACSIM will be able to view agent actions and play a certain part of an agent inside the scenario. This means that the agents will have to have some sort of GUI because otherwise the user will not notice their actions. If they have received or sent a report, then its GUI must also be showing those, for information purposes.

Architecture

A global overview of the MACSIM system is given in Figure 3. These are all the components that can be distinguished from a global perspective.

Simulation component: In this component a simulation of the concerned area can be given. The physical properties on a location (x,y,z) on time t can be read, and in this version of the program this means for instance fire, gas dispersion and explosions. For the gas dispersions formulae that are being used in commercial available software are used, the Gaussian Plume Model, to be exact. The simulation will be simplified to reduce computing complexity and design issues, because creating a scientifically valid simulation could be a full master project on its own, and as the purpose of this system was to create a design proof of concept, a slimmed-down version of the simulation will be available, to prevent the development from this component to overshadow the development of the reasoning engine.

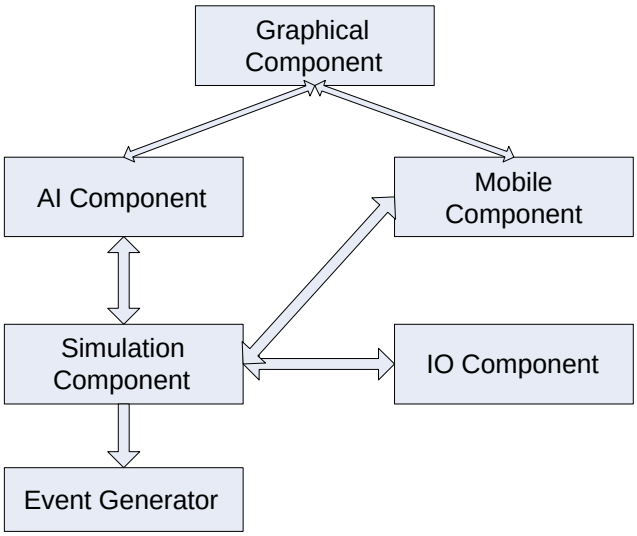


Figure 3: Component Architecture

Event Generator: This event generator should be able to generate crisis events, which is an XML-based script. When the script unfolds, events are being launched and those events have their effect on the simulation environment. This event generator in the future should work in two ways: first it generates events from a predefined script; secondly it should be able to generate new events as the participants in the simulator respond to the system. Based on the time schedule it will be decided which ones of these two forms of scripting will be available in the first version of MACSIM.

Graphical component: The graphical component of this program will consist of several user interfaces. One user interface will be used to setup the scenario, and simulation parameters. Another Interface will be used as a representation of what is going on at the crisis center. It will also contain a graphical representation of the area that will show the incoming reports of the people that are currently in the simulation area.

AI-component: This component will consist of a knowledge based system that, based on decision rules that are derived from first-hand experience from experts and real-life complaints from people that smell gases, helps in deciding what probably is the most realistic scenario that is currently happening. This is of critical importance in the first stages of the development of a crisis, when not much information is known and a first hypothesis can be made through a knowledge-based system. Inside the knowledge based system the agent has several hypotheses available. These hypotheses are represented as frames or sets of properties and rules and actions. Each moment that new knowledge arrives, on a scoreboard the scores for each different frame are placed. The frame that is triggered the most is the frame or hypothetical event that is chosen as most probable event. This could mean that also information is coming in that is conflicting with the hypothesis, but as long as the information rejecting the current hypothesis is not convincing enough, the current hypothesis is being maintained. If there is enough conflicting information to reject the hypothesis, then another more probable hypothesis frame is chosen. Therefore a different set of knowledge will become available and this means also different actions that will be performed by that specific agent (see Figure 4).

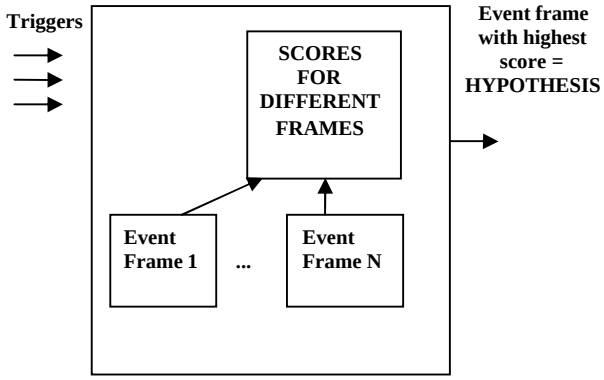


Figure 4: Frame-based Hypothesis Forming

Mobile component: In the future, the crisis response components should also be able to be used in real life (no simulation) and for this, a mobile component should be created that can be run on a handheld pc, that enables user participants to create reports based on an icon based application such as ISME [14]. These reports are then treated in the same fashion as other reports, and can be used in the system in the same way to help determining what the most probable crisis situation my be.

I-O component: This component will take care of the regular file IO operations that might be necessary inside the system, such as loading and saving files. Examples could be loading and saving of scripts, or saving or saving graphical representations while running.

IV. IMPLEMENTATION

The design for MACSIM was made in Java and it was designed in such a way that it was:

- ***Expandable***: Being a research object, the system should be able to incorporate new features that could improve the functionality of the system but are currently not yet in the scope of the system. In the design we should take care that we design as 'open' as possible, to facilitate future work on the system.

- ***Not memory consuming***: The agents that are at a certain location must get the data they need as fast as possible, so their observing and interpreting of the data does not get delayed.

- ***Without too much hard coded data***: Being a simulation, the system relies on a lot of mathematical, physical and empirical constants. It should be avoided to include those constants in java code, because this would mean that if they should be changed for some reason, that people have to dig in the Java source to change whatever they want. It is more preferable to read constants from text or XML files, where they are being imported by the system and can be edited easily for future use.

- ***Operating on JADE***: With JADE being one of the leading platforms for multi-agent programming, it was decided that the simulation should run on JADE, to facilitate running of system on mobile devices in the future.[5]

In Figure 5 MACSIM is being described as a series of Java software components that are independent parts of the system and exchange information during the course of a simulation. In the crisis center component in Figure 5 the reasoning mechanism Figure 4 is used. In some of these components external open source software components like JADE and Jess were used. In the description of each component the use of that particular tool will be described.

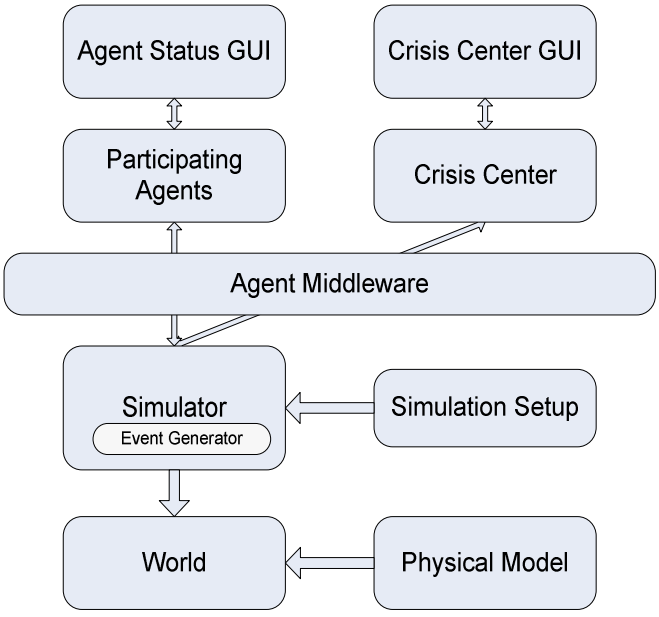


Figure 5: Software Component Decomposistion

Simulator: This is the main component of the program, where the simulation is being set up and run. The startup and all necessary parameter checking are done here. From here all the other necessary components are started. The simulator ensures that the agents receive the waypoints that the agents need to correctly observe the world. By passing the essential data via the agent middleware ensures the agents in the field receive the appropriate data.

World: the simplification of the world for which we are making simulations. We cannot incorporate everything that is in the real world into our world model. Because of complexity and performance reasons we make the world in which a scenario is developing as simple as possible. The World consists of waypoints which contain the values that the agents observe.

Physical Model: The collection of models that enable the events to impose changes inside the world. These are mainly a set of formulae that can be changed or added when necessary. The formulae are designed in such a way that they can be replaced by more advanced counterparts very easily.

Event Generator: This component makes sure that events, that are part of a scenario, are executed in the right way. This means that the event generator takes care of updating the world model in the fashion that a particular event requires. It uses the formulae available in the Physical Model to update a particular event. There are as many Event Generators as there are active events in the simulator. Whenever according to the scenario a new event needs to be launched, a new event generator is started that takes care of updating the waypoints with data relevant to that event.

Simulation Setup: In this window all the properties of a certain simulation can be edited, loaded and saved. After the user is satisfied with the scenario, it can be stored and executed. Things like starting time, ending time, number of participating agents can be set up in this GUI. It is really up to the creativity of the programmer and the complexity of the simulation presented how much parameters and details can be set.

Agent Middleware: The agent middleware ensures that the messages that are sent between agents are being processed correctly. Also the updates of the world are processed so that the agents have them at their disposal. MACSIM uses JADE as Agent middleware, which is a very popular platform for the programming of multi-agent systems.

Participating Agents: These are the key elements in the system. They are the main characters in the simulation and as the scenario unfolds, the characters respond to the changing situation. The agents can communicate with each other about the current status, and they can use reasoning for making decisions about what they should do the next. Participating agents can also contact the crisis center. For communicating, they use JADE agent messaging, through FIPA compliant ACL Messages [5]. These messages contain information that is based on a Java-Based crisis ontology developed by Siska Fitrianie [2]. For reasoning they use a knowledge base. As a knowledge base Jess is being used for its easy interaction with Java objects, such as the ontology elements inside Fitrianie's crisis ontology.

Crisis Center: during a crisis, a crisis center is usually being set up to coordinate the crisis response effort. Here all the information about the crisis comes in and is being interpreted by the people who are there. The crisis center interprets all the messages that are coming in and based on rule-based reasoning, they create orders for their subjects that can be sent by using agent messages. They issue commands to the participating agents that are in the field, based on the information that they get from all available sources. This information is being stored inside something that is called an interpreted world. This is an interpretation of what the crisis center believes is going on inside the world. In real life, the interpreted world is usually some sort of real or digital map onto which status information is being added. Based on what they see on the map, the crisis team decides what to do. The decision making functionality described in Figure 4 is implemented in this component.

Interface during simulation (Agent Status GUI and Crisis Center GUI): During the simulation the agents are making all kinds of decisions, and a lot of communication is exchanged as well. We want to keep track during run-time of what is going on in the world, so we need some way to eavesdrop on the communication during a crisis. In quite the same way that communication during a flight is being recorded, the agent communication will be shown on a user interface while the simulation is running. Besides that, for the agents that the user identifies with during the simulation, agents can be controlled and agent actions can be initiated from this GUI. For the participating agents, the status of the agent is being shown. As a visual component of the crisis center, an icon-based crisis map will be used, that is able to show incoming crisis messages on top of a map of a crisis area. This was made using components from another application called 3MNews, designed by Iulia Tatomir [4].

V. EVALUATION

For testing MACSIM, simulations had to be run, and for simulations we needed scenarios. These scenarios are script-based. This means that a scenario consists of an XML-based script. A script is a combination of a time and an event with the source location of the event added to it. In this way the event generator knows exactly what to do to update the world based on that event.

To test the simulation we had to define some test scripts to evaluate the system. At this point the simulation consists of automated predefined actions based on the observation of effects of script-generated predefined events. Later on this will evolve into a more dynamic form of scripting, in which game players can initiate actions themselves and scenarios can be changed as a result of those actions.

The first minutes of an example test scenario could look like the contents of Table 1:

Table 2: First minutes of Example Scenario

Time	Event + Location	Agent	Possible Action
14:00	Fire starts at (X,Y)	General Public	Run Away + Report trough GSM or SMS
		Sensor	Activation smoke alarm + reporting to Crisis Center
		Crisis Center	Sends a call to Fireman about developing situation.
		Professional (Fireman)	Receives a call from Crisis Center to go to (X,Y) and is on his way
14:02	Fire develops	General Public	Complaint to Crisis Center about developing smoke
		Professional (Fireman)	While on its way to (X,Y) receives info from crisis center that fire is developing
14:04	Explosion	General Public	Report to Crisis Center about damage and casualties and a loud bang
		Crisis Center	Order extra units of Firemen and Police to (X,Y)
14:05		Professional (Fireman)	Arrives at (X,Y) and starts extinguishing fire
		Professional (Policeman)	Receives order from Crisis Center to seal off crisis area around (X,Y)

VI. CONCLUSION

With the development of MACSIM we have devised a system that is able to simulate crisis situations based on scripts. For this purpose a crisis generator component, a communication layer and intelligent agents were developed and tested and although further testing is still needed, first result look very promising.

An advantage of this system is that it enables us to add new physical models, new intelligent agents and new types of crisis

events very easily; also existing components can be updated very fast.

As an open simulation platform, MACSIM offers a lot of room for further development. The first additions planned for the near future are the use of gaming software for the creation of graphical components for 3D graphics and more sophisticated algorithms for agent reasoning, like Bayesian belief networks. When more features become available, there will inevitably also be a need to visually represent these new features. Therefore the existing user interface will be upgraded to improve the visual experience and sense of presence for the user that runs the simulation. At this point the user interface just shows message traffic, but real time crisis simulation of course is something that cannot do without a realistic visual representation.

Furthermore, the design offers enables adding more advanced reasoning algorithms. In all cases where new intelligence/ knowledge has to be added or intelligence/ knowledge should be updated, it is important to have discussions with the domain experts during the development process to get useful first-hand experience and information that can be represented in rules, behaviors and agent actions. The rules in this way will also become better and more sophisticated.

REFERENCES

[1] Ernest Friedman-Hill. Jess in Action. Manning, 2003.

[2] S. Fitrianie and L.J.M. Rothkrantz. The represenation of the world. Technical report, TU Delft, 2006.

[3] L.J.M. Rothkrantz. Interactive Simulation in Disaster Management (ISDM). 2006.

[4] I. Tatomir. 3MNews - Message-based Multimodal News. Technical report, TU Delft, 2006.

[5] Fabio Bellifemine, Agostino Poggi, and Giovanni Rimassa. Jade: a fipa2000 compliant agent development environment. pages 216–217, 2001.

[6] EPA and NOAA. ALOHA User's Manual, 2004.

[7] SAFER Systems Inc, SAFER REALTIME/TRACE Technical Manual.

[8] Hazmat: Hotzone project website, (http://projecthazmat.org)

[9] The Simulator project website, (http://www.ics.uci.edu/~projects/drillsim/)

[10] COMBINED Project Website, (http://combined.decis.nl)

[11] B. Tatomir, L. Rothkrantz and M. Popa, "Intelligent system for exploring dynamic crisis environments", Third International Conference on Information Systems for Crisis Response and Management ISCRAM 2006, May 2006.

[12] P. Klapwijk L.J.M. Rothkrantz, "Topology Based Infrastructure for crisis situations", Third International Conference on Information Systems for Crisis Response and Management ISCRAM 2006, May 2006.

[13] S. Mehrotra, C. Butts, D. Kalashnikov, N. Venkatasubramanian, K. Altintas, Haimin Lee, J. Wickramasuriya, R. Hariharan, Y.Ma: CAMAS: A Citizen Awareness System for Crisis Mitigation. University of California, Irvine

[14] Paul Schooneman. ISME - Icon based System for Managing Emergencies. Master's thesis, TU Delft, 2005.