Universidade do Porto

Faculdade de Engenharia

FEUP

Multi-Agent System for an Airline Operations Control Centre

António Pedro da Silva Mota

Supervisor: Luís Paulo Reis (PhD)

Co-Supervisor: António Castro (MSc)

April 2008

Faculty of Engineering, University of Porto
Informatics Engineering Department
Master in Informatics and Computing Engineering

Universidade do Porto
Faculdade de Engenharia
FEUP

Multi-Agent System for an Airline Operations Control Centre

António Pedro da Silva Mota

Supervisor: Luís Paulo Reis (PhD)
Co-Supervisor: António Castro (MSc)

April 2008

Abstract

Being planned in advance, the operation of an airline company is often subject to change. This is due to problems with crew members, aircraft, or passengers, which affect the beginning of a flight at the time planned. The Operational Control Centre (OCC) is the entity that manages the operation of an airline company in the moments preceding the flights. It has the main goal of solving any problems that might occur at the lowest possible impact and cost, in a process called disruption management.

The management of the operation, which covers the activities of monitoring, event detection, and troubleshooting, is usually still a highly manual process, heavily based on the tacit knowledge of the various members of the OCC. The main goal of this project is to implement a multi-agent system that automates various activities on this process, increasing the quantity and quality of the information available to the entity responsible for making the changes to the plan.

For this purpose, it was developed an agent that monitors the operation of the airline in a particular base, allowing the visualization of the flights that will arrive, that have already arrived, that will depart, and that have already departed from that base. It was also created an event detection mechanism, which recognizes existing problems based on the customizable definition of events given by agents. Finally, the problem resolution is achieved by a number of specialist agents, each implementing a different resolution method based on Artificial Intelligence.

The objectives of this project were achieved, as the system was implemented and tested in an airline company. The implemented system makes it easier for the events to be detected, and achieves better solutions to the problems than the previous system.

Resumo

Sendo planeada com antecedência, a operação de uma companhia aérea é frequentemente alvo de alterações. Tal se deve a problemas com tripulantes, aviões, ou passageiros, que afectam a realização de um voo na hora planeada. O Centro de Controlo Operacional (CCO) é a entidade que gere a operação de uma companhia aérea nos momentos que antecedem a realização dos voos, tendo como principal objectivo solucionar eventuais problemas com o menor impacto e custo possível, num processo designado por Gestão de Irregularidades Operacionais.

A gestão da operação, que abrange as actividades de monitorização, detecção de eventos, e resolução de problemas, é um processo essencialmente manual, fortemente baseado no conhecimento tácito dos vários membros do CCO. O principal objectivo deste projecto é implementar um sistema multi-agente que automatize essas várias actividades, aumentando a quantidade e a qualidade de informação disponível à entidade responsável por efectuar as alterações ao planeamento.

Para o efeito foi desenvolvido um agente que monitoriza a operação da companhia aérea numa determinada base, permitindo identificar os voos que irão aterrar, que já aterraram, que irão partir, e que já partiram, dessa base. Foi também criado um mecanismo de detecção de eventos, que reconhece os problemas existentes com base na definição parametrizável, dada por agentes, do que é um evento. Finalmente, a resolução do problema é efectuada por uma série de agentes especialistas, cada um implementando métodos de resolução distintos baseados em Inteligência Artificial.

Os objectivos do projecto foram alcançados, tendo o sistema sido implementado e testado numa companhia aérea real. Verifica-se que o sistema desenvolvido possui uma maior facilidade em detectar problemas e permite um acréscimo na qualidade das soluções obtidas relativamente ao sistema anterior.

Acknowledgements

I would like to thank my supervisor, Prof. Luís Paulo Reis, without whom this project would never see the light. His advices and orientation were invaluable.

I'm very grateful to Eng. António Castro, project manager at TAP Portugal, for all the help he gave me. I would also like to thank my colleagues at TAP Portugal.

I'm very grateful to my family for all their support.

Finally, I would like to thank all my friends; you know who you are...

Table of Contents

Table of Figures

Table of Tables

Glossary

Aircraft Event	A situation that might affect the schedule of the aircraft (for example, an aircraft malfunction or a flight delay).
Aircraft Recovery	The process of solving problems related with aircrafts and triggered by aircraft events.
Cabin Crew Member	A crew member that provides on-board service to passengers.
Cockpit Crew Member	A technical crew member responsible for piloting the aircraft.
Crew Event	A situation that might affect the schedule of the crew member (for example, a crew member delayed, or a crew member that does not report for duty).
Crew Recovery	The process of solving problems related with crew members and triggered by crew events.
Operational Base	A company base with flights, crew members and aircrafts, and with a schedule of flights that start and ends at the base.
Operations Controller	A person that works in the Operations Control that monitors and changes the schedule of an aircraft and/or a flight.
Operations Manager	A person that works in the Operations Control that manages and takes decision regarding the schedule of an aircraft and/or a flight.
Passenger Event	A situation that can affect the passengers of a flight.
Passenger Recovery	The process of solving problems related with passengers and triggered by passengers events.
Perdiem Value	A value that is given to crew members, for each day of work, to pay for food or other expenses when they are out.
Stand by Crew Member	A crew member scheduled so that he can be used to replace another crew in a flight.

1 Introduction

This chapter sets the frame of the project, explains the reasons behind its development, establishes its objectives and expected results, and presents the structure of the rest of this document.

1.1 Overview

The disruption management of an airline operation is a process by which problems that might affect the proper implementation of the operational scheduling of the company (problems related with crew members, aircraft, and/or passengers of a flight), are solved. This process is based on three key components: monitoring (supervising the airline operation in a particular base), event detection (identification of situations that could put at risk the operational planning), and resolution of problems (identify solutions that can mitigate the problems encountered). Therefore, the proper functioning of this system is essential so that the operation of the airline company can take place with regularity and, as far as possible, within the plan previously established.

Since this is a very important process for the airline, it usually has a department responsible for its implementation: the Operational Control Centre (OCC). In this centre, operating 24 hours a day, working groups of people are responsible for monitoring the operation, detecting events, and propose solutions to the Supervisor of the OCC, an entity that decides whether the solution will be implemented or not.

Those solutions are achieved based mostly on the tacit knowledge of the people and there is no automated manner, nor even a formal documented one, of how to address the resolution of the problems. And though there are software tools that help the elements of the OCC throughout the process (especially at the stage of monitoring and when obtaining information useful to solve the various problems), they are often obsolete, do not cover the whole process, and are not integrated.

This project lies in the area of Artificial Intelligence, and it consists on the development of a Multi-Agent System (MAS) representing the OCC and the various elements existing there. The resolution of the problems will be made by a group of agents, a process known as disruption management. These agents use meta-heuristics, although the system is designed to permit the inclusion of an unlimited number of agents that can use any method of problem resolution.

1.2 Motivation

This project was developed in a real airline company, and the motivation to develop it came from the observation of the needs of the OCC of that company. Indeed, it was found that the tools that assist the monitoring of the company operation were not the most intuitive, that the detection of events had a weak computerized component, requiring an repetitive effort to the elements of the OCC, and that the resolution of problems was mainly based on tacit knowledge of those elements.

It was thus clear that there was a good opportunity to develop a system that could focus on these situations and that in some way could catch and use the knowledge of the elements of the OCC, to make the process automatic, or at least to automate some of its more repetitive tasks.

Although there is extensive research in the area of MAS applied to problems of scheduling on airlines, there are very few works of really functional MAS on real airlines, capable of an integrated resolution of the type of problems that are found on the OCC. Therefore, this project would be innovative, useful not only for the company in question, but potentially also for others.

Finally, it is important to highlight the economic value that comes from the use of a system like this. First, by automating part of, or even all, the process, it is no longer required the presence of so many elements in the OCC, which lets the company reduce costs with staff; this reduction is even more evident because the OCC is a department that operates 24 hours a day. Furthermore, the use of Artificial Intelligence methods to resolve the various problems can contribute to the improvement of the quality of the solutions implemented, which translates into a decrease in costs associated with the modification of the operation of the airline company.

In a few words, this project has been developed based on the needs of a real airline, in the absence of similar deployments in the field of aviation, and in the economic value that such deployment could bring to the company.

1.3 Objectives

The multi-agent system to be developed should monitor the operation that is being conducted on each operational base of the company (that is, the flights, the crew members who have reported for duty, etc.), detecting events that may correspond to problems that need to be resolved (some minor events can be skipped). Some of the possible events are: aircraft failures, flight delayed due to bad weather and / or congestion of air traffic, shortages of crew members, etc. The system should permit the definition of what is an event (a ten minutes delay of a crew member can be a problem for one airline, but not for another).

The system must have a set of agents, cooperating with each other, that are specialists in the resolution of the various types of problems. It is hoped that the presence of agents that implement different methods for solving the same problem, will increase the robustness of the system because of this redundancy.

The best solution should be submitted to the Supervisor of the OCC, through an agent that represents him.

Finally, the system should have a visual interface that allows the supervisor to monitor the operation, to detect the events, and to observe the solution proposed for the specific problem at hand.

1.4 Expected Results

With the deployment of the system, it is expected that:

1. The time taken to detect an event with the system will be less than the time achieved without its use. That is a consequence of the process of monitoring the operation in search for events becoming automatic.

2. The time taken to start the problem resolution with the system will be less than without it. That's because the specialist agents are always ready to solve the problem (humans have always a set up period, besides the sometimes difficult switch between problems), and the process of communicating that problem to them is fast and reliable.

3. The information gathered during the problem resolution process with the system will be more complete than without it. That's because the system is able to programmatically access the systems where that information lies, which is faster and more reliable than the access through a user interface.

4. As a consequence of (3), the time taken to obtain a specific solution with the system will be faster than without it (or, given a fixed amount of time, a process using the system will generate more solutions than a process without it).

5. The solution found with the system will be based on more heterogeneous processes than without it. That's because the system is open to the use of any number of specialist agents, and each one can implement a completely different resolution method.

6. Following (4) and (5), the solution found with the system will be better than without it; that is, the use of the system will considerably decrease the cost associated with the disruption management process.

1.5 Document Structure

The next chapter analyses the problem that led to the development of the project, and presents in detail how the operation of an airline is managed at the Operational Control Centre. Finally, the problem is decomposed in several sub problems.

Then, it is presented the state of the art of the areas in which the project lies, and of the technologies it uses: it will be examined the disruption management process, the optimization problems, several meta-heuristics, the multi-agent systems, and the framework used to develop the system - JADE.

The fifth chapter describes the work done, from the specification of the project, to the implementation.

Then, there is the analysis of the results obtained with the use of the system, and a case study is presented.

Finally, the conclusions of the project are discussed and improvements for possible future developments are purposed.

2 Problem Description

This chapter analyses the problem that led to the project, presenting in detail how the operation of an airline is managed at the Airline Operational Control Centre (AOCC). Then, it decomposes the main problem in several sub problems. The chapter ends with a discussion about the recommended references to this chapter.

2.1 Disruption Management at AOCC

The AOCC is the area of an airline company that manages the airline operation during a specific time window, usually within the operation day. That process is performed by several people, each playing one of the following roles (Kohl, et al., 2004 p. 5):

- Aircraft Control: Manages the aircraft resource, and is often the central coordination role in operations control. It is divided in long and short hauls flights.
- Crew Tracking: Manages the staffing of flights, assuring that every flight has all the necessary crew members to take off.
- Aircraft Engineering: Deals with the unplanned service and maintenance of the aircraft, and with the short term maintenance scheduling.
- Customer Service: Ensures that passenger inconvenience cause by the implementation of a specific solution is taken into account in the decision process.

The schedule of the airline operation is defined during the airline scheduling process, depicted in Figure 1 (Kohl, et al., 2004 p. 2).

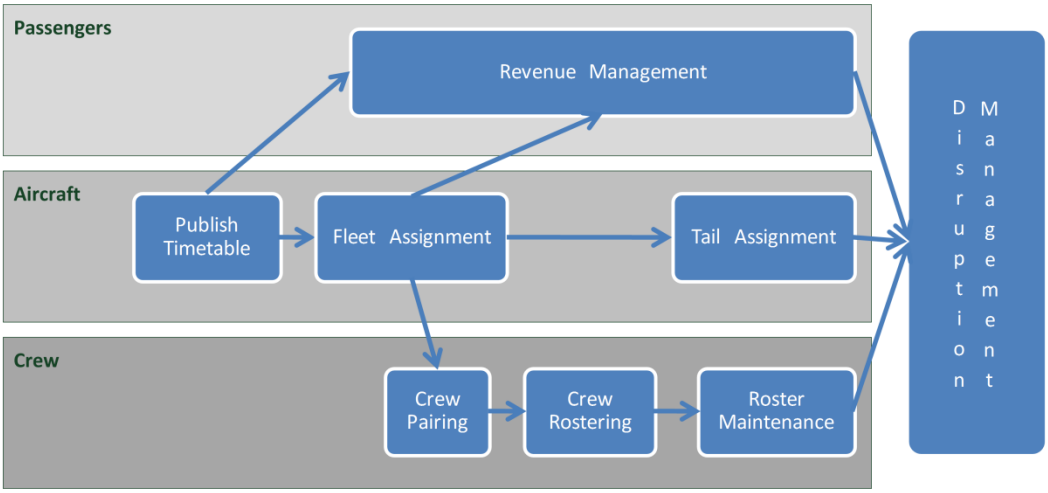

Figure 1 - Scheduling Process Phases

There are three relatively independent aspects that must be managed (Castro, 2006 pp. 25, 26):

- **Passengers**. The revenue management starts after the timetable publication, and lasts until the operation day. It consists of adjustments of prices and seat availability as a response to the market.

- **Aircraft**. The aircraft management starts with the publishing of the schedule of flights that can be sold to passengers, which triggers the revenue management and the allocation of fleets to those flights. The latter, in turn, prompt the allocation of aircrafts to flights (tail assignment) and the crew pairing phase.

- **Crew**. After the fleet assignment, anonymous pairings are constructed. Then (usually a few weeks before the operation day), those pairings, as well as other activities like stand by duties, training, reserves, or days off, are assigned to individual crew members. Finally, later changes (for instance, commercial flight alteration or crew availability change) are dealt with in roster maintenance.

At last, when one enters within the time window managed by AOCC, all the problems related with aircrafts (like a delay or a malfunction), crew (for instance, a crew member that didn't report for duty), or passengers (usually a consequence of the other two) are managed in the disruption management (also called operations recovery) phase.

The disruption management process is presented in Figure 2.

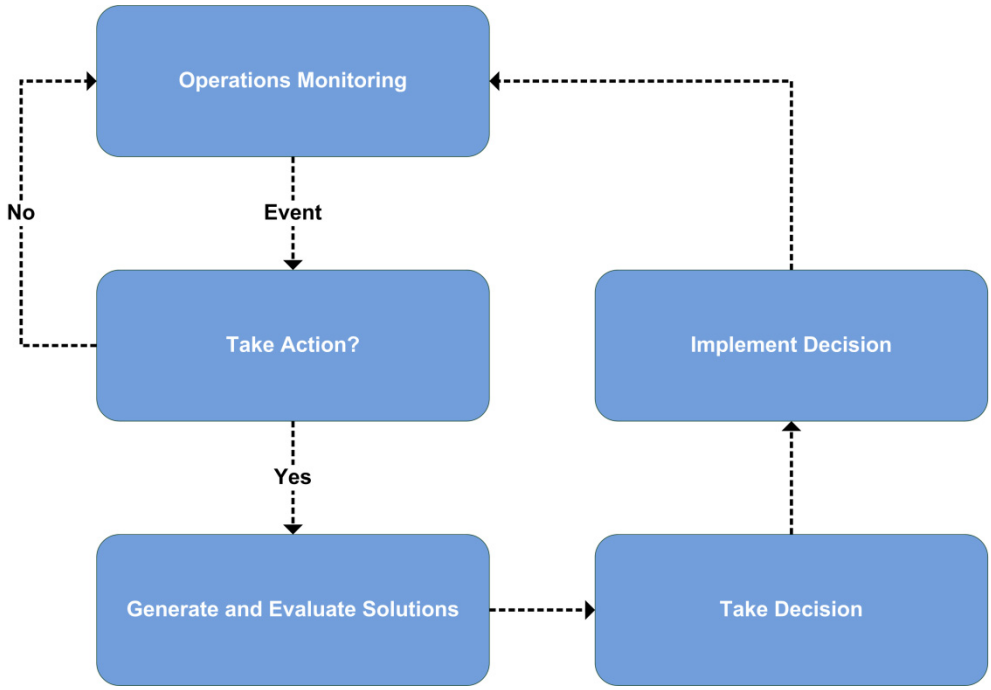

Figure 2 – Disruption Management Process

The personnel at AOCC are continually monitoring the airline operations, in search for an event. When an event happens, a decision must be made whether that event is relevant or not (for instance, a one minute delay of a flight usually is not an event that requires action). If the event is considered relevant, then a number of solutions are generated and evaluated, and

eventually one will be accepted as the final decision. At last, that decision is implemented (that is, the change to the operation schedule is made).

According to (Kohl, et al., 2004 p. 8), the evaluation of solutions in the disruption management process at AOCC is guided by the following objectives:

- Get passengers and their luggage to their destination on-time with the booked service level.

- Minimize the real costs including excess crew costs, costs of compensation, hotel and accommodation to disrupted passengers and crew, and tickets on other airlines.

- Get back to the scheduled plan as soon as possible.

The same authors consider that the main roles for computer systems in disruption management are (1) process the large amounts of operational data and present potentially critical events for a human, and (2) generate and evaluate possible solutions for a specific problem.

2.2 Sub problems

The main problem can be broken down into several sub problems. That is, to implement a system that automates the disruption management process, the following things must be done:

- **Monitor the airline operations**. What is the necessary information to monitor the airline operations? How to present that information so that the operation is easily apprehended by the OCC personnel?

- **Detect Events**. What is an event? What are the types of events? How to make its definition flexible? How to integrate the monitoring and the event detection? How and where to signal an event?

- **Find Solutions**. What is a problem? What is the necessary information to find solutions to that problem? How the OCC personnel solve the problems? What methods can be used to solve the problem? How to integrate the event (problem) detection and the finding of solutions to that problem? How to present the solutions to the OCC personnel?

- **Evaluate Solutions**. How can the solution quality be measured? What are the factors that define a good solution? How to make the definition of those factors flexible? How take into account all the stakeholders affected by an operational change?

In addition to these technology-related problems, the development of the system in a real airline company was conditioned by the following hurdles:

- **Complexity of the company's database.** The database that holds the company information regarding its operations is complex, and must be thoroughly analyzed so that the information that the systems needs can be correctly obtained.

- **Domain language.** The requirement analysis and the database study require a deep knowledge about the technical terminology of aviation. Therefore, that language must be mastered so that the communication with the system stakeholders can be effective, and so that the sources of data are well understood.

2.3 References

(Kohl, et al., 2004) offers an introduction to airline disruption management, provides the readers with a description of the planning processes ,and delivers a detailed overview of the numerous aspects of airline disruption management. In addition, the authors report on experiences from a large research and development project on airline disruption management.

(Castro, 2006) proposes the designing of a multi-agent system for monitoring and operations recovery for an airline operations control centre.

2.4 Summary

This chapter has analyzed the problem that led to the project, presenting in detail how the operation of an airline is managed at the AOCC. The disruption management process was framed in the airline scheduling process, and the three aspects that must be managed in the former process – passengers, aircraft, and crew - were presented. This notion (that there are several stakeholders affected by a plan change) is very important and will be present throughout the document.

The disruption management process was presented, which is the base for the algorithm of the resolution manager agent described in chapter 4.

In addition, the main problem was decomposed in several sub problems, which turned the specification of the solution (chapter 4) more manageable.

Having presented the problem, it is time to analyze the state of the art of the areas in which the project lies, and of the technologies it uses.

3 State of the Art

This chapter presents the state of the art of the areas in which the project lies, and of the technologies it uses. To that purpose, it will examine the disruption management process, the optimization problems, several meta-heuristics that can be used to solve these problems, the concepts of agents and multi-agent systems, and the framework used to develop the system – the Java Agent DEvelopment Framework (JADE).

3.1 Disruption Management

The operation recovery problem has usually been solved through Operations Research techniques, either by focusing on a specific type of problem (aircraft recovery, crew recovery, or passenger recovery), or by taking a more integrated and general approach that tackles all the problems.

Regarding the **general approaches**, the Descartes (DEcision Support for integrated Crew and AiRcrafT recovery) research project ran over 3 years from 2000 to 2003 in a jointly cooperation between several airline companies and a university. The main objective of the project was to develop a disruption management system based on a holistic approach. The system should integrate the decisions of the different resources in one integrated feasible decision. The main focus was on the four key resources of the problem namely aircraft, flight and cabin crew, and passengers.

In relation to **aircraft recovery**, the solution of the problem consists of delaying flights (preventing a flight, on purpose, from departing on time), cancelling flights (assign no aircraft to a flight), and/or swapping aircraft to flight assignments (switch the aircraft of a flight by another one), in order to create a more preferable revised flight schedule. According to (Love, et al., 2001):

- Delays are preferable to cancellations.
- Some flights have a higher priority than others.
- Reassignments of aircraft to flights in the immediate future are not preferable if the disruptions of the schedule can be resolved by making reassignments further down the schedule.
- Sometimes, some specific aircraft must end at a certain airport for scheduled maintenance.
- It is a high priority that the number of changes made is kept to a minimum.

Regarding **crew recovery**, the solution of the problem consists of delaying crew (which causes the delay of the flight), using stand by crew (swap the problematic crew member with one that is in standby), swapping crew to flight assignments (swap the problematic crew member with one that is assigned to another flight), or use extra-crew (transporting crew as

passengers to flights at other airports where a crew shortage exists). According to (Love, et al., 2001):

- As many as flights as possible should have the necessary crew assigned.
- Crew assignments should not cause flight delays.
- Swapping crew to flight assignments is preferable to using standby crew.
- When swapping crew or using standby crew, some crews are more preferable than others.
- It is a high priority that the number of changes made is kept to a minimum.

Finally, the **passenger recovery** problem is usually addressed by focusing on the other two problems (aircraft and crew recovery), as the passenger problems can be minimized if those problems are solved.

3.2 Optimization Problems

An optimization problem is the problem of finding the best solution from all feasible solutions. More formally, an optimization problem $P = (S, f)$ can be defined by:

- A set of decision variables $X = \{x_1, x_2, \dots, x_n\}$
- The domains of the variables $D_1, D_2, \dots, D_n$
- A set of constraints among the variables that define the joint values that the variables may take.
- The objective function $f : D_1 \times D_2 \times D_n \rightarrow R^+$ to be minimized or maximized.

  where the solution space S is the set of all feasible solutions $S = \{s = \{(x_1, v_1), (x_2, v_2), \dots, (x_n, v_n)\} \mid v_i \in D_i,\ s \text{ satisfies all the constraints}\}$.

The goal of the optimization problem is to find an instance of the solution space, $s \in S$, for which the objective function has the minimum or maximum value.

If the search space is very small, one may exhaustively evaluate the objective function on each point in the search space and select the optimal solution. However, for real problems that is usually impracticable because of the huge size of the search space. Therefore, the essence of optimization techniques lies in the determination of the optimal or of a sub-optimal instance of the search space, without actually having to search the entire search space. This is accomplished by choosing a starting point, and then altering one or more variables in an attempt to reach a better point. The way the search space is explored is defined by the meta-heuristic that the problem resolution uses.

3.3 Meta-heuristics

(Hillier, et al., 2005) defines an heuristic method as a procedure that is likely to discover a very good feasible solution, but not necessarily an optimal one, for the specific problem being considered. The procedure is often an iterative algorithm, where each iteration involves conducting a search for a new solution that might be better than the best solution found previously. The algorithm usually finishes after reaching a reasonably solution, or after a certain amount of time as passed. The main problem about heuristic methods is that they were created to address a specific problem. Therefore, a lot of effort is needed to adapt the method to solve a (sometimes slightly) different problem.

In recent years, high level heuristics – meta-heuristics – were created that can be applied to a diversity of problems. A meta-heuristic is a general solution method that provides a general structure and a number of strategy guidelines in order to explore the solution space wisely.

This section will analyze three meta-heuristics: hill-climbing, steepest ascent hill climbing, and simulated annealing.

3.3.1 Hill-Climbing

From a (randomly) selected start point in the search space (that is, a solution), a step is taken in a random direction. If the fitness of the new generated point is greater than the one of the previous point, than it becomes the new solution; otherwise, the solution remains the same. The process is repeated until the hill-climbing algorithm no longer accepts any steps from the trial solution.

The algorithm is easy to implement, but is inefficient and gives no protection against finding a local minimum rather than the global one. More specifically, (Russel, et al., 1995) points out that:

- The algorithm can reach a local maximum. A local maximum is a peak that is higher than its neighbor's solutions but is not the highest value of the objective function. The algorithm will stop at these points.

- The algorithm can reach a plateau. A plateau is an area where the objective function is flat. The algorithm will search erratically in these areas.

- The algorithm can reach a ridge. A ridge is an area with a point where, even without it being a local maximum, all available moves will make the solution worse. Ridges depend on the method chosen to calculate neighbor solutions.

One way of addressing these limitations is to repeat the process many times with different starting points.

Figure 3, taken from (Russel, et al., 1995), depicts the hill-climbing algorithm.

```
function HILL-CLIMBING(problem) returns a state that is a local maximum
  inputs: problem, a problem
  local variables: current, a node
                   neighbor, a node

  current ← MAKE-NODE(INITIAL-STATE[problem])
  loop do
     neighbor ← a highest-valued successor of current
     if VALUE[neighbor] ≤ VALUE[current] then return STATE[current]
     current ← neighbor
```

Figure 3 – Hill-Climbing Algorithm

3.3.2 Steepest Ascent Hill-Climbing

The steepest ascent hill-climbing is a variation of the hill-climbing algorithm. While the hill-climbing algorithm tests all the neighbors of the current solution to choose the best, the steepest ascent hill-climbing accepts the first neighbor whose fitness is greater than the one of the current solution. As a consequence, the steepest ascent hill-climbing usually converges quicker than the hill-climbing.

3.3.3 Simulated Annealing

In simulated annealing (Kirkpatrick, et al.), from a (randomly) selected start point in the search space (that is, a solution), a new solution is generated. If the fitness of the generated solution is greater than the one of the previous, the generated solution becomes the new solution. If the fitness is not greater, the solution still has a probability P of being accepted:

$$P = \exp(-\Delta E \div T)$$

ΔE is the increase in the cost function that would result from the step. T is the temperature, a parameter that measures the tendency to accept the current candidate to be the next solution if this candidate is not an improvement on the current solution: if T is high, the optimization procedure is free to accept many varied solutions, even if they don't represent improvements, but as it drops, this freedom diminishes. The value of T is initially set high and is periodically reduced according to a cooling schedule that usually is as simple as:

$$T_{t+1} = \alpha T_t$$

Figure 4 represents the flowchart of the simulated annealing algorithm.

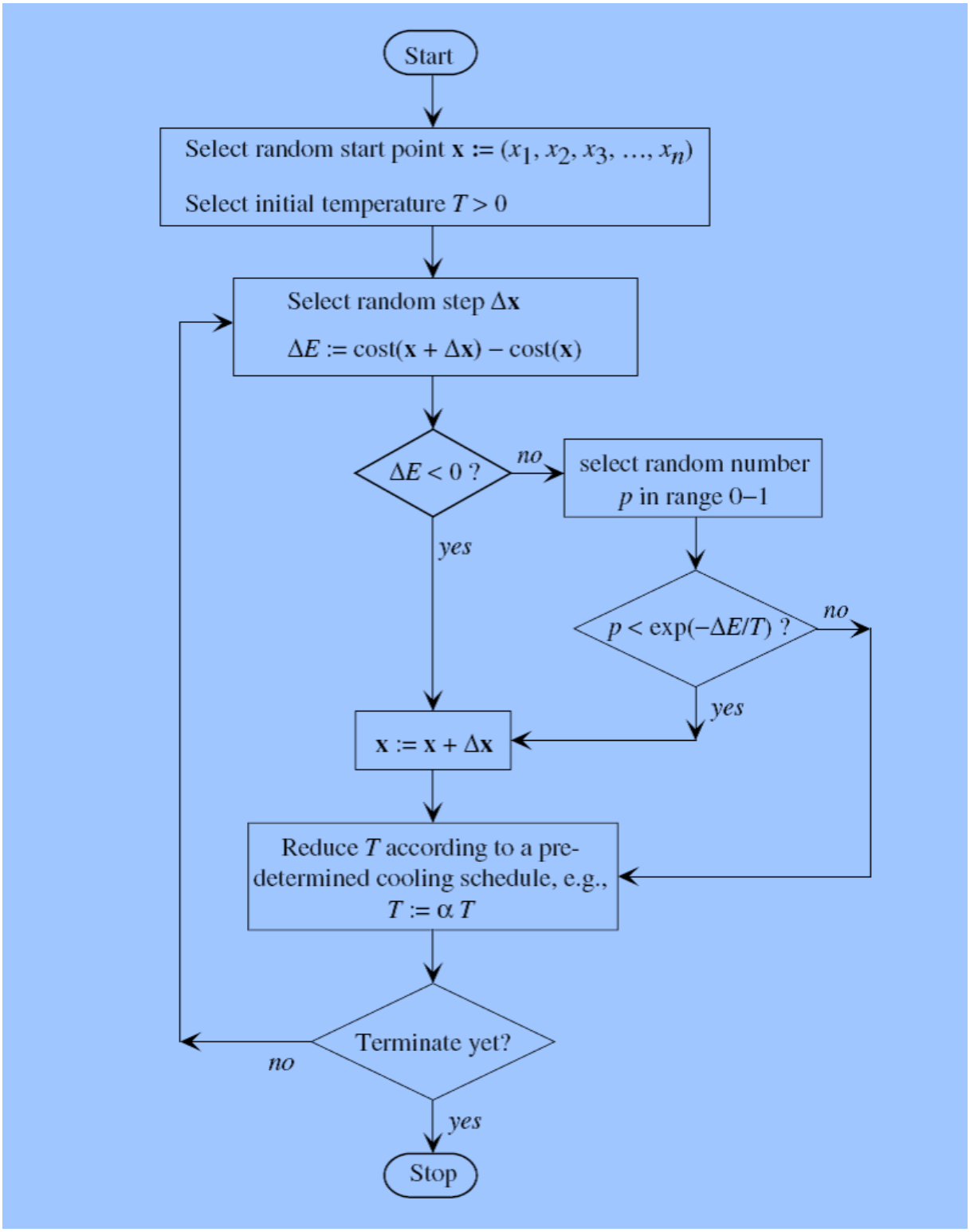

Figure 4 – Flowchart of the Simulated Annealing Algorithm

Additionally, the algorithm can reduce T only after a fix number of iterations have been produced at each temperature.

(Hillier, et al., 2005) advises that the following details need to be worked out to fit the structure of a specific problem, before applying the algorithm to that problem:

- How should the initial trial solution be selected?
- What is the neighborhood structure that specifies which solutions are immediate neighbors (reachable in a single iteration) of any current solution?
- How to randomly select one of the neighborhoods of the current solution to became the current candidate to be the next solution?
- What is an appropriate temperature schedule?

3.4 Agents and Multi-Agent Systems

According to (Russel, et al., 1995), an agent is essentially a special software component that provides an interoperable interface to an arbitrary system and/or behaves like a human, working for some clients in pursuit of its own agenda.

An agent has the following characteristics (Wooldridge, 1996):

- Autonomy: it operates without the direct intervention of humans and has control over its actions and internal state.
- Social: it cooperates with humans or other agents in order to achieve its tasks.
- Reactive: it perceives its environment and responds in a timely fashion to changes that occur in the environment.
- Proactive: it does not simply act in response to its environment but is able to exhibit goal-directed behavior by taking initiative.

Despite being usually implemented as an object, there are significant differences between agents and objects. (Wooldridge, 1996) points out the following ones:

- Agents embody a stronger notion of autonomy than traditional objects, and they decide for themselves whether or not to perform an action that was requested by another agent.
- Agents are capable of flexible behavior, and the standard object model has nothing to say about such types of behavior.
- A multi-agent system is inherently multi-threaded, in that each agent is assumed to have at least one thread of control.

According to (Wooldridge, 1996), an agent-based solution is appropriate in the following scenarios:

- The environment is open, or at least highly dynamic, uncertain, or complex. Systems capable of flexible autonomous action are often the only solution in such environments.

- Agents are a natural metaphor for the problem. For instance, the AOCC is naturally modeled as a society of agents that cooperate with each other to solve the various problems that occur.

- Distribution of data, control, or expertise. In some environments (like the AOCC), the distribution of either data, control, or expertise means that a centralized solution is difficult.

- Legacy systems. One way of solving the problems caused by obsolete systems is to wrap the legacy components, providing them with agent-layer functionality, and enabling them to communicate and cooperate with other software components.

3.5 JADE

JADE (Java Agent DEvelopment Framework) is a platform implemented in the Java language that simplifies the development of multi-agent systems, especially in terms of management of agents, communication and debugging. Figure 5 shows the components of this platform.

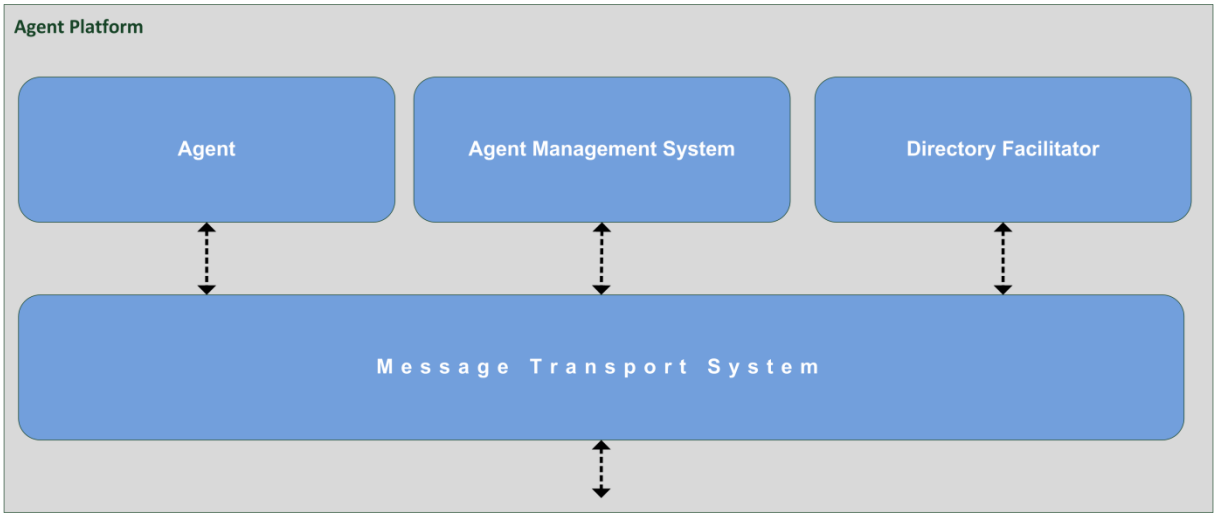

Figure 5 – JADE Platform Components

The platform is composed by the agents, the management system of the agents (Agent Management System - AMS), the directories searcher (Directory Facilitator - DF) and the system that transports the messages that are sent between agents (Message Transport System - MTS).

The AMS has the task of overseeing the access and use of the platform and of maintaining a directory of agents' identifiers (AID) and its states; that is, it provides a service of names. This component is unique to each platform and all the actors register in it.

The DF provides a service similar to the yellow pages, in which an agent may seek other agents based on the service they provide.

The MTS handles communication between agents. This communication is done using the ACL language using one of three methods: event passing (if within the same container), RMI (if between containers from the same platform) or IIOP Protocol (if between different platforms).

Figure 6, taken from (Bellifemine, et al., 2007), shows the relationships between the main architectural elements of JADE.

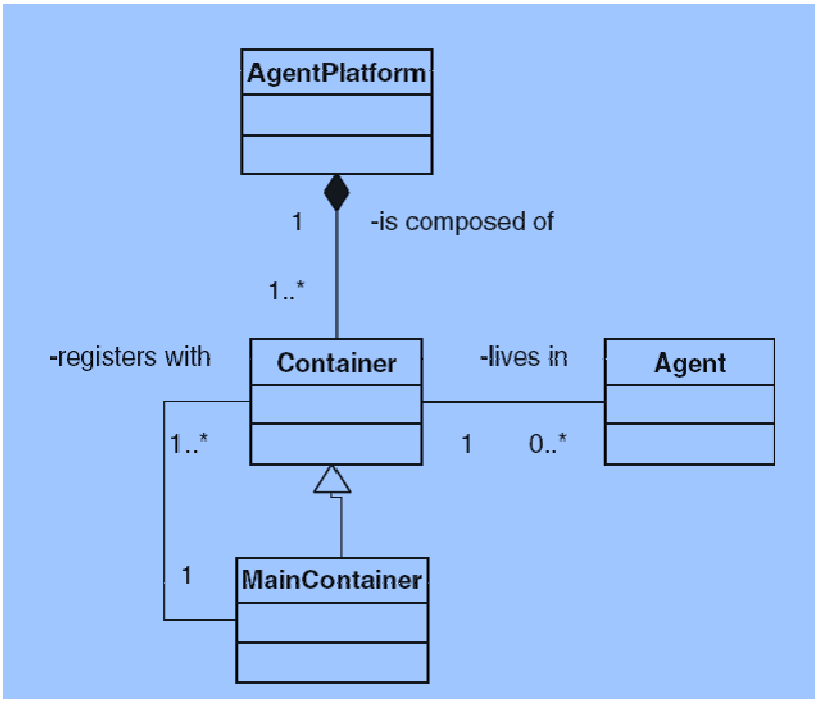

Figure 6 – Relationship between the Main Architectural Elements of JADE

The JADE platform has a number of interesting features, of which stand out:

- Platform according to FIPA (Foundation for Intelligent Physical Agents) specification, using ACL messages for communication between the agents that compose the multi-agent system;
- Distributed platform, in which agents may reside in different, and geographically separated, machines;
- Support for the mobility of agents, enabling the transferring of both his state and its code;
- Several tools with graphical interface which assist the development and testing of the system;
- Support for the execution of multiple activities of the agent through the use of the behavioral model;

- Support for ontologies.

3.6 References

Regarding the Descartes project, (Kohl, et al., 2004) reports on the experiences obtained during the research and development of the project, presents the current mode of dealing with recovery, and describes the results of the first prototype of a multiple resource decision support system.

(Rosenberger, et al., 2001) deals with aircraft recovery, proposing a model that addresses each aircraft type as a single problem. The problem is formulated as a Set Partitioning master problem and a route generating procedure. The objective is to minimize cost of cancellation and retiming, and the parameters definition lies in the responsibility of the controllers. To solve the master problem in suitable time, a heuristic is used to select only a subset of aircraft to be involved in the Set Partitioning problem.

(Bratu, et al., 2006) presents an integrated recovery approach to solve the operation recovery problem, and has a strong emphasis on reducing passenger arrival delays.

(Abdelgahny, et al., 2004) addresses the flight crew recovery problem for an airline with a hub-and-spoke (a air transportation system where local airports offer air transportation to a central airport where long-distance flights are available) network structure. The crew recovery problem is divided in four sub-problems: misplacement problems, rest problems, duty problems, and unassigned problems. The most used techniques to solve the problem are delaying, swapping, use of extra-crew, and use of stand by crew.

(Wikipedia, 2008) offers a simple description about optimization problems.

More detailed information can be found in (Nocedal, et al., 1999) and (Yang, 2008).

(Hillier, et al., 2005) gives an introduction to operations research, and has a chapter on meta-heuristics. (Wikipedia, 2007) and (Wikipedia, 2008) describe, respectively, the simulated annealing and hill-climbing meta-heuristics. A more comprehensive description of these meta-heuristics can be found in (Russel, et al., 1995).

(Wooldridge, 1996) gives an introduction to the field of agent based computing. It provides a comprehensive introduction to intelligent agents and multi-agent systems.

More information about the JADE system can be found at the project's homepage (JADE Development Group, 2008).

(Caire, 2003) is a JADE programming tutorial for beginners, useful when using this framework for the first time.

A more advanced guide about programming in JADE can be found in (Bellifemine, et al., 2007), and about administrating the framework in (Trucco, et al., 2007).

(Bellifemine, et al., 2007) is a book describing JADE.

3.7 Summary

This chapter presented the state of the art of the areas in which the project lies, and of the technologies it uses. The several approaches used to tackle the operation recovery problem were described, establishing the various options that the agents have to solve a problem related with an aircraft or a crew member.

Because the problem is an optimization one, the three meta-heuristics (hill-climbing, steepest ascent hill-climbing, and simulated annealing) that are used by the specialists agents described in chapter 4 were analyzed.

Because this is a multi-agent system, some information regarding this topic was presented, and the reasons behind the use of agents were explained.

Finally, the chapter ended with the description of the JADE platform, which simplifies the development of multi-agent systems.

4 Solution Specification and Implementation

The previous chapters described the problem and the state of the art in the areas where the project lies. This chapter gives a description of the specification and implementation of the system.

The chapter is divided into ten sections. It starts with the description of the overall architecture of the system, which is useful to understand the relationships between the subsystems that are described in later sections. Then, it describes the data structures used in the system. After that, the specification and implementation of each subsystem (monitoring, event detection, and resolution subsystem) is analyzed. To conclude, the solution generation and evaluation is analyzed, as well as the communication between the different subsystems.

4.1 System Overview

Figure 7 shows the overall architecture of the multi-agent system.

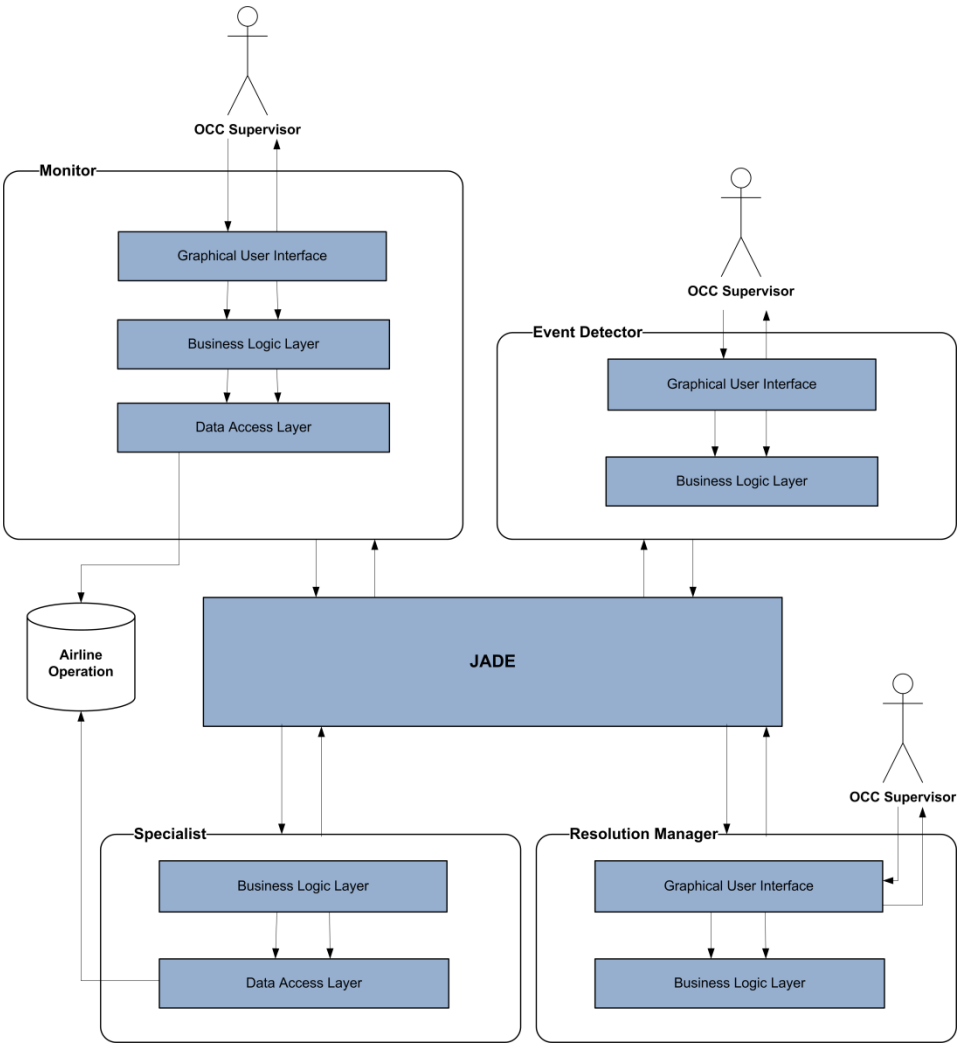

Figure 7 – Overall Architecture of the Multi-Agent System

There are four types of agents:

- **Monitor**, which monitors the operation of the airline company in a specific time window and in a specific base. This usually is accomplished by accessing databases that store real time information about the airline company operation. The monitoring subsystem is described in sections 4.3 and 4.4.

- **Event Detector**, which defines the types of events that must be detected and presented as either warnings or problems. The event detection subsystem is described in sections 4.5 and 4.6.

- **Resolution Manager**, which receives a problem and manages the resolution in cooperation with the specialist agents. The resolution subsystem is described in sections 4.7 and 4.8.

- **Specialist**, which is responsible for the resolution of a problem using a specific method. The two specialist agents implemented are described in sections 4.7 and 4.8. The generation and evaluation of each solution that they generate during the resolution process is described in section 4.9.

The communication between the agents is done through the JADE system and is described in section 4.10.

Additionally, the figure shows the existence of a data store, that has information about the airline company operations. That data store is accessed by the monitor and by the specialist agents.

The figure also shows the presence of a human: the OCC supervisor. This element interacts with:

- The monitor agent, to see the details of the operation of the airline company, and to order the resolution of a specific problem that was triggered by an event.

- The event detector agent, to define the types of events that must be detected during the monitoring process.

- The resolution manager, to establish some parameters that are used in the resolution process, and to see the final solution given by the system.

Figure 8 shows a typical sequence of utilization of the system, focusing on the interaction between agents/humans.

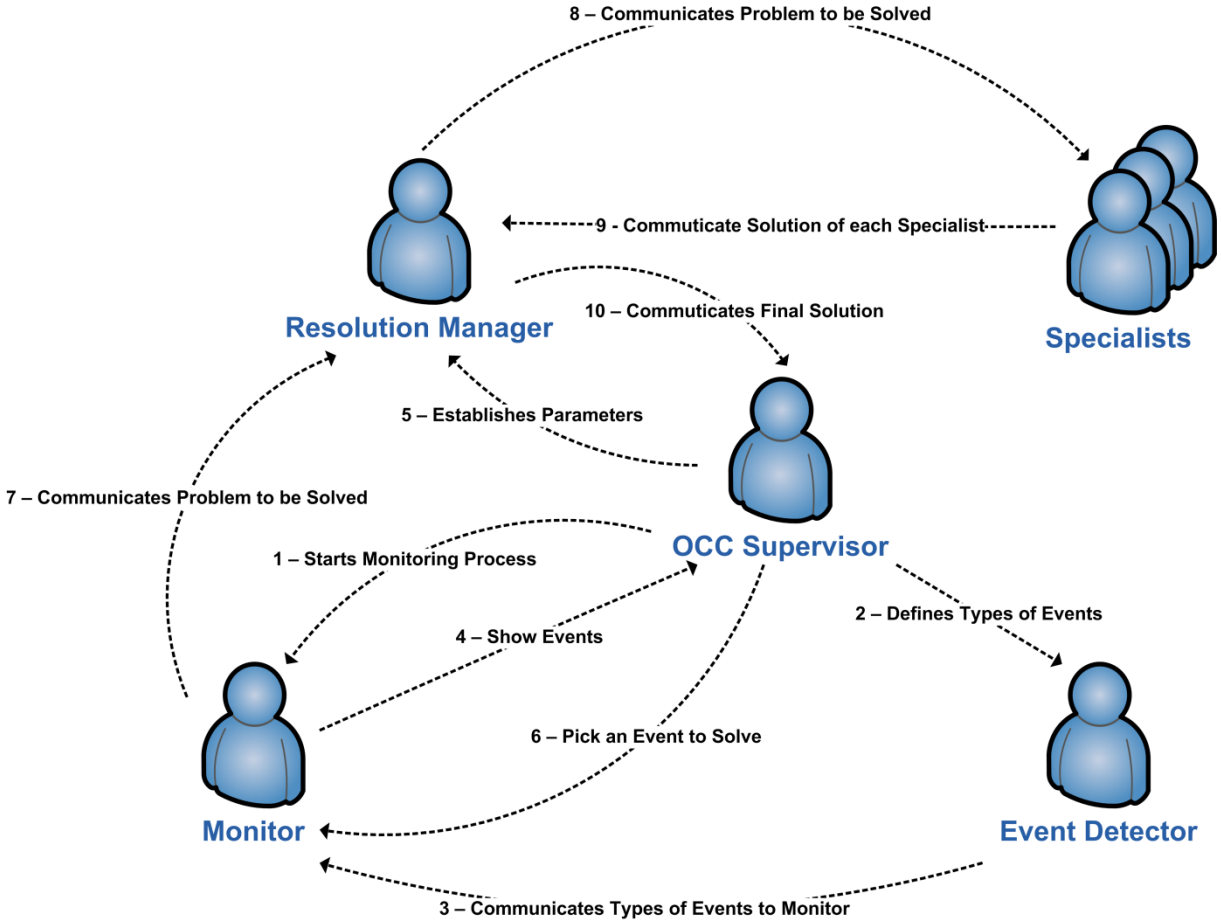

Figure 8 – Typical Sequence of Events

The OCC supervisor starts the monitoring process and defines the types of events that are to be considered warnings and problems. Then, the event detector agent communicates those types of events to the monitor, which shows the warnings and/or the problems to the OCC supervisor. This entity establishes the parameters of the resolution process, and decides to solve a specific problem. That problem is communicated by the monitor to the resolution manager, which in turn communicates it to all the specialist agents. After arriving at a solution, each agent sends it to the resolution manager, which picks the best and shows it to the OCC supervisor.

Figure 9 shows all the use cases of the system. The JADE actor is represented because the reception of messages from other agents is periodically checked by using a type of JADE behavior (so, the JADE actor periodically asks the system to check for new messages, acting as an actor).

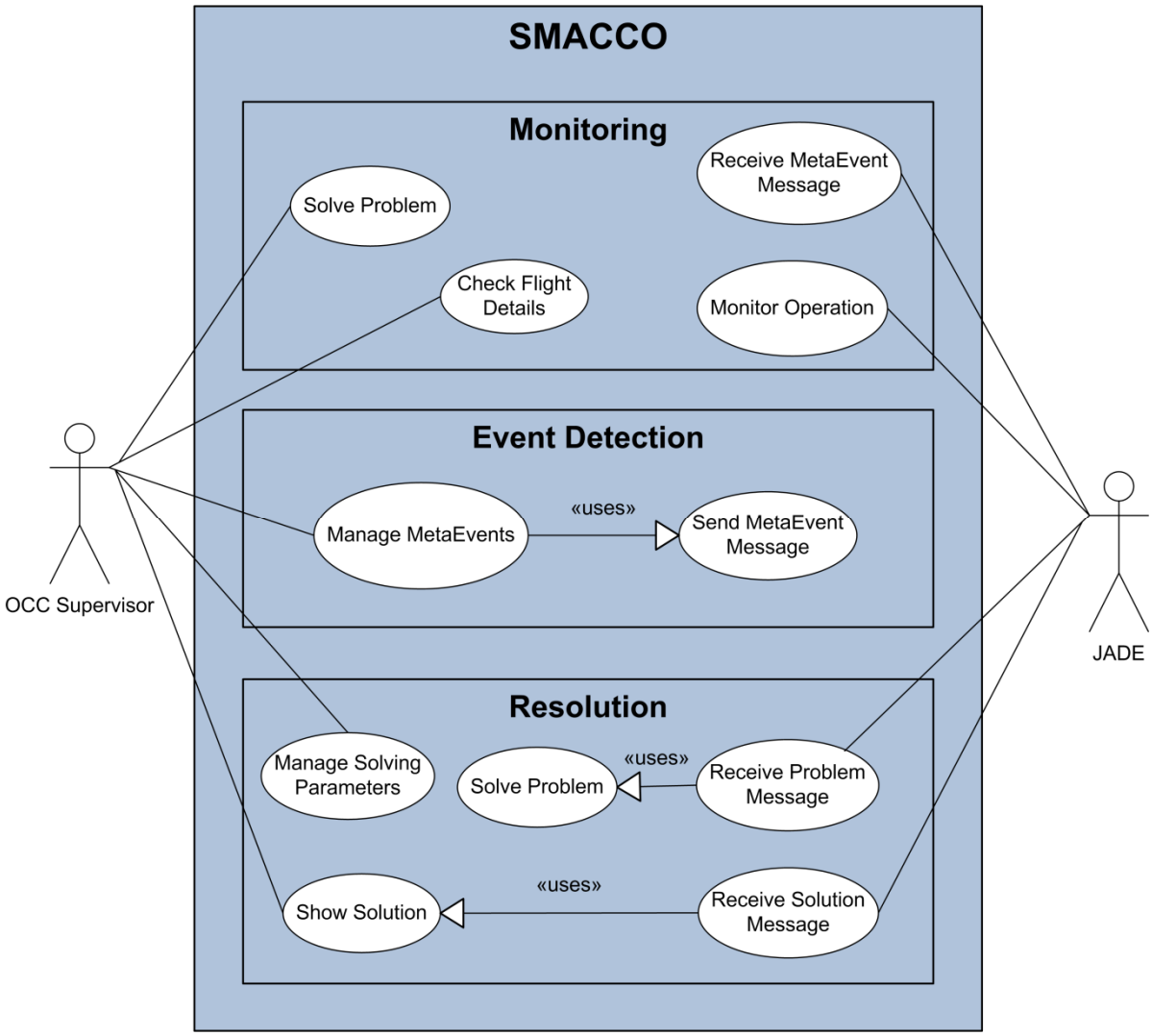

Figure 9 – System Use Cases

In the monitoring subsystem, the OCC supervisor can check the details of a particular flight that is being monitored, as well as order the resolution of a specific problem. The JADE system periodically asks the system to monitor the operation and to check the arrival of MetaEvent messages.

In the event detection subsystem, the OCC supervisor can manage MetaEvents, that is, add, remove, or modify a type of event. Each time the set of MetaEvent is changed, a message is sent to the monitor.

Finally, in the resolution subsystem, the OCC supervisor can manage the solving parameters, and check the solution. The JADE system periodically asks the system to check the arrival of problem messages (which triggers the resolution of that problem), and solution messages (which triggers its presentation to the OCC supervisor).

4.2 Data Structures

The system uses several data structures to pass information between each layer of the agent, and also to pass information between agents. Figure 10 shows the class diagrams of these data structures.

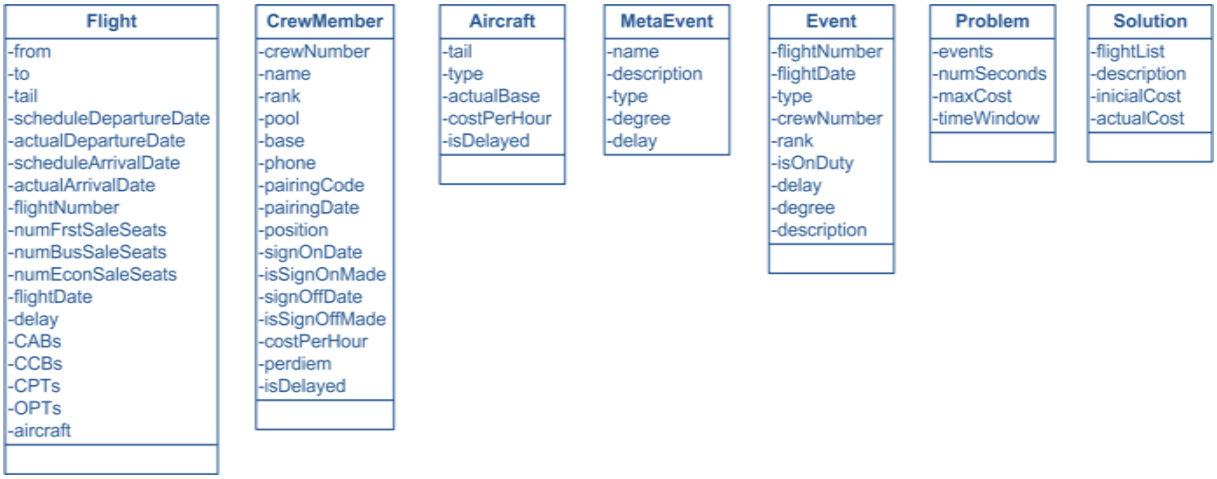

Figure 10 – Data Structures

A MetaEvent defines the types of events to be detected. If a problem triggered by one or more events is to be solved, an object of type Problem is created. The resolution process creates objects of type Solution. The monitor and resolution subsystems use objects of type Flight, which in turn has an Aircraft and several CrewMembers associated.

4.3 Monitoring Subsystem Specification

Table 1 describes the use case "Monitor Operation".

Use Case ID:	SMACCO-BUC001.00
Use Case Name:	**Monitor Operation**
Created By:	António Mota
Creation Date:	10/12/2007
Actors:	JADE
Trigger:	This use case is initiated when the JADE cyclic behavior triggers its action (the operation monitoring).
Description:	1. Obtain flights that are going to arrive within a specific amount of time. 2. Obtain flights that arrived within a specific amount of time ago. 3. Obtain flights that are going to depart within a specific amount of time. 4. Obtain flights that departed within a specific amount of time ago. 5. Signal the flights on the interface, separating the short and long-haul flights. 6. Check if there are types of events to monitor 7. If there are types of events to monitor, check what flights (if any) match the conditions of the type of event. 8. If there are flights that correspond to the type of event conditions, signal the type of event in the interface.

Table 1 – Use Case "Monitor Operation"

Table 2 describes the use case "Check Flight Details".

Use Case ID:	SMACCO-BUC002.00
Use Case Name:	**Check Flight Details**
Created By:	António Mota
Creation Date:	15/12/2007
Actors:	OCC Supervisor
Trigger:	This use case is initiated when the OCC supervisor clicks on a flight.
Description:	1. Obtain the detail about the crew members of the flight. 2. Obtain the detail about the aircraft of the flight. 3. Show those details on the interface

Table 2 - Use Case "Check Flight Details"

Table 3 describes the use case "Receive Meta Event Message".

Use Case ID:	SMACCO-BUC003.00
Use Case Name:	**Receive Meta Event Message**
Created By:	António Mota
Creation Date:	12/12/2007
Actors:	JADE
Trigger:	This use case is initiated when the JADE cyclic behavior triggers its action (checking if there are any meta event messages).
Description:	1. Refresh the types of events list so that the list of types of events received in the message is incorporated.

Table 3 – Use Case "Receive Meta Event Message"

Table 4 describes the use case "Solve Problem".

Use Case ID:	SMACCO-BUC004.00
Use Case Name:	**Solve Problem**
Created By:	António Mota
Creation Date:	12/12/2007
Actors:	OCC Supervisor
Trigger:	This use case is initiated when the OCC supervisor decides, by clicking on a button, to solve a problem.
Description:	1. Prepare the message to be sent to the resolution subsystem. a. Obtain the descriptor of the agent that manages the resolution process. b. Set the list of event objects (see figure 10) as the content of the message. 2. Send the message.

Table 4 – Use Case "Solve Problem"

Figure 11 presents the methods that are used in the monitoring subsystem.

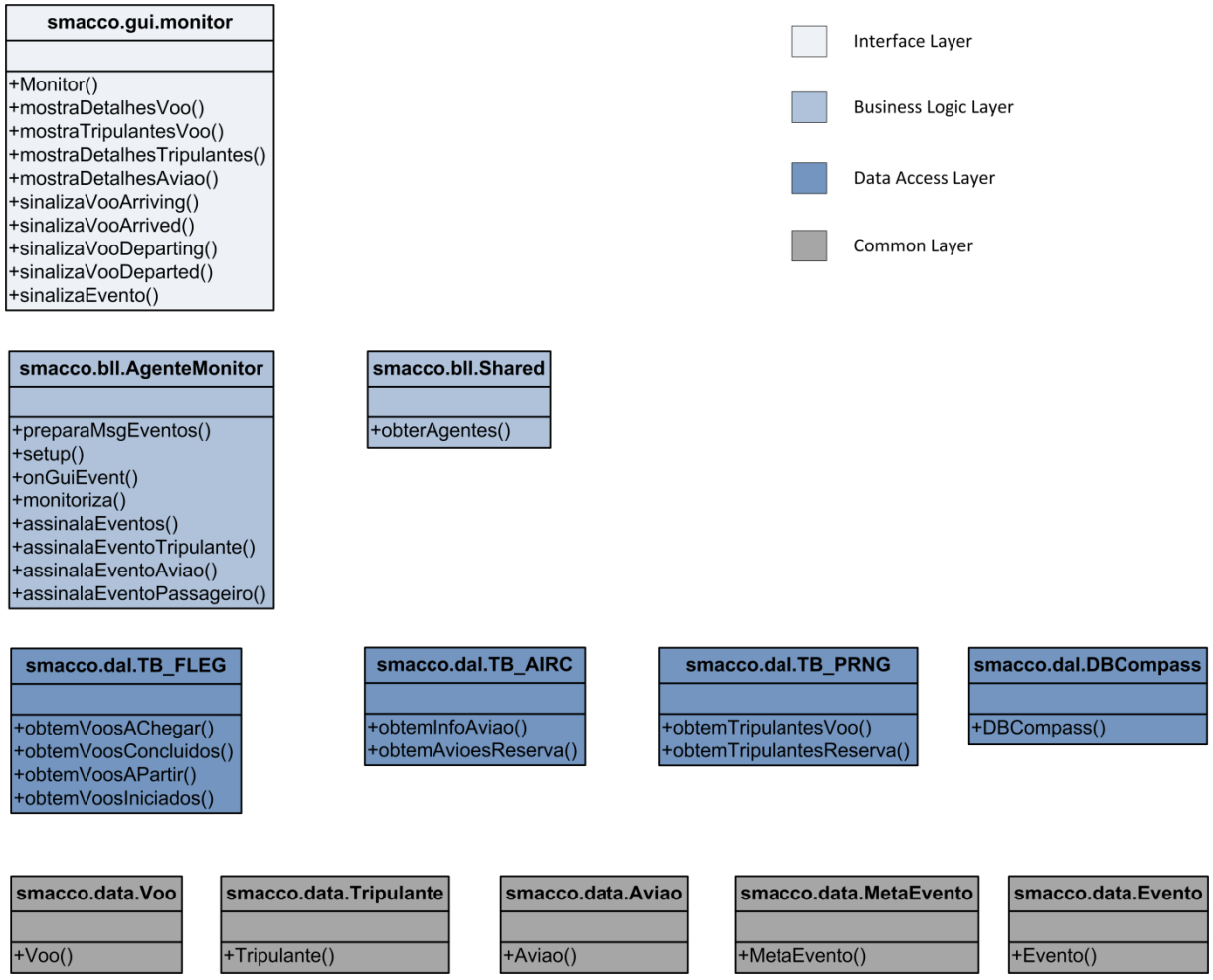

Figure 11 – Methods Used in the Monitoring Subsystem

The monitor agent implements five behaviors:

- A *TickerBehaviour* that periodically accesses the company's operational database to monitor the operation.
- A *CyclicBehaviour* that waits for a meta event message to arrive, using a *ReceiverBehaviour* to filter the appropriate messages.
- An *OneShotBehaviour* that sends a message with a set of events to be solved.
- A *ParallelBehaviour* that execute the former behaviors concurrently.

4.4 Monitoring Subsystem Implementation

Figure 12 shows the *behaviourMonitor* class, which extends *TicketBehaviour*. The method *onTick* contains the piece of code that will be executed periodically.

```
class BehaviourMonitor extends TickerBehaviour
{
    public BehaviourMonitor(Agent agent, long period)
    {
        super(agent, period);
    }

    @Override
    protected void onTick() {
        try
        {
            monitoriza("NB");
            monitoriza("WB");
            if(definicoesEventos.size() > 0)
                assinalaEventos();
        }
        catch (SQLException ex) {ex.printStackTrace();}
    }
}
```

Figure 12 – Class BehaviourMonitor

The agent periodically monitors the narrow body and wide body airline operation and, if there are any types of events to be detected, checks if any flight is under the conditions necessary to consider itself a warning or a problem.

Figure 13 shows the *BehaviourResolucaoProblema* class, which extends *OneShotBehaviour*. The piece of code inside the *action()* method will be executed only once.

```
class BehaviourResolucaoProblema extends OneShotBehaviour
{
    public BehaviourResolucaoProblema(Agent agente)
    {
        super(agente);
    }

    @Override
    public void action()
    {
        ACLMessage msgAEnviar = preparaMsgEventos(eventos, Shared.obterAgentes(super.myAgent, "AgenteManager"));
        send(msgAEnviar);
    }
```

Figure 13 – Class BehaviourResolucaoProblema

In this behavior, a message is sent from the monitor agent to the resolution manager containing the list of events that triggered the problem.

4.5 Event Detection Subsystem Specification

Table 5 describes the use case "Manage Meta Events".

Use Case ID:	SMACCO-BUC005.00
Use Case Name:	**Manage Meta Events**
Created By:	António Mota
Creation Date:	19/12/2007
Actors:	OCC Supervisor
Trigger:	This use case is initiated when the OCC supervisor adds, removes, or alters, a type of event.
Description:	1. Refresh the data model so that the alterations of the user are reflected.

Table 5 – Use Case "Manage Meta Events"

Table 6 describes the use case "Send Meta Event Message".

Use Case ID:	SMACCO-BUC006.00
Use Case Name:	**Send Meta Event Message**
Created By:	António Mota
Creation Date:	30/12/2007
Actors:	OCC Supervisor
Trigger:	This use case is initiated when the OCC supervisor adds, removes, or alters, a type of event.
Description:	1. Prepare the message to be sent to the monitor subsystem. a. Obtain the descriptor of the agent that monitors the operation. b. Set the list of meta event objects (see figure 10) as the content of the message. 2. Send the message.

Table 6 – Send Meta Event Message

Figure 14 presents the methods that are used in the event detection subsystem.

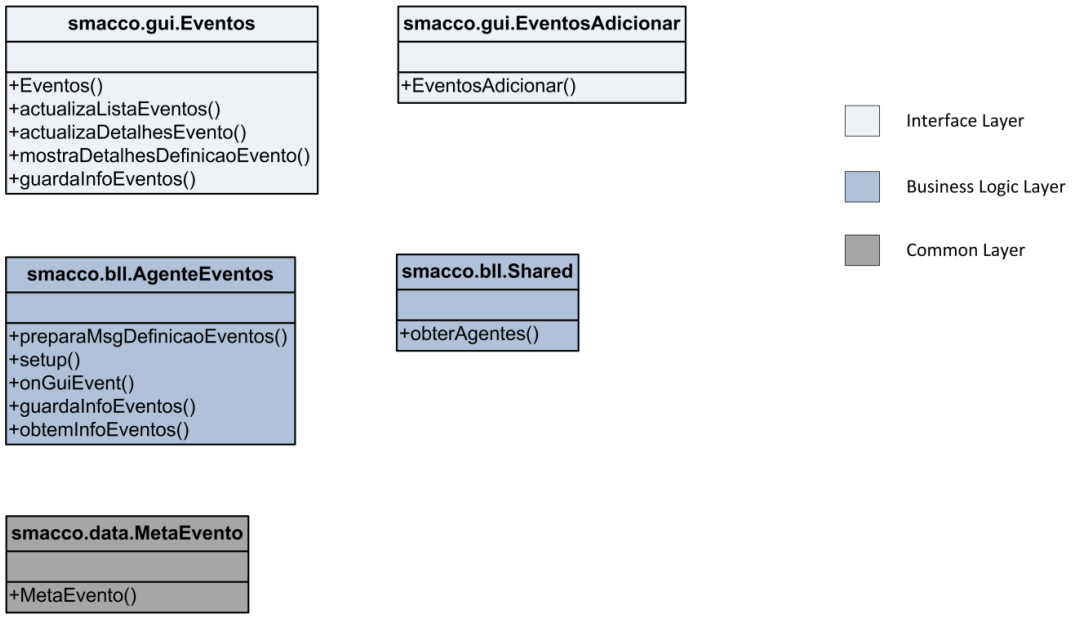

Figure 14 - Methods Used in the Event Detection Subsystem

The event detection agent implements one behavior: an *OneShotBehaviour* that sends a message with a set of types of events to be monitored.

4.6 Event Detection Subsystem Implementation

Figure 15 shows the *BehaviourGestorEventos* class, which extends *OneShotBehaviour*. The piece of code inside the *action()* method will be executed only once.

```
class BehaviourGestorEventos extends OneShotBehaviour
{
    public BehaviourGestorEventos(Agent agente)
    {
        super(agente);
    }

    @Override
    public void action()
    {
        ACLMessage msgAEnviar = preparaMsgDefinicaoEventos(definicoesEventos, Shared.obterAgentes(super.myAgent, "AgenteMonitor"));
        send(msgAEnviar);
    }

    private ACLMessage preparaMsgDefinicaoEventos(ArrayList<MetaEvento> definicoesEventos, ArrayList<AID> destinatarios)
    {
        ACLMessage mensagem = new ACLMessage(ACLMessage.INFORM); // performativa Inform

        for(int i=0; i<destinatarios.size(); i++)
            mensagem.addReceiver(destinatarios.get(i));

        try
        {
            mensagem.setContentObject(definicoesEventos);
        }
        catch(IOException e)
        {
            System.out.println("Erro ao estabelecer o conteúdo da mensagem de definição de eventos");
            return null;
        }

        return mensagem;
    }
}
```

Figure 15 – Class *BehaviourGestorEventos*

This behavior sends a message from the event detector agent to the monitor agent with the definition of the types of events that are to be detected during the monitoring process carried out by the latter agent.

4.7 Resolution Subsystem Specification

Table 7 describes the use case “Manage Solving Parameters”.

Use Case ID:	SMACCO-BUC007.00
Use Case Name:	**Manage Solving Parameters**
Created By:	António Mota
Creation Date:	27/12/2007
Actors:	OCC Supervisor
Trigger:	This use case is initiated when the OCC supervisor alters a parameter to be used in the problem resolution process.
Description:	1. Refresh the data model so that the alterations of the user are reflected and the new parameters are used in the problem resolution process.

Table 7 – Use Case "Manage Solving Parameters"

Table 8 describes the use case "Receive Problem Message".

Use Case ID:	SMACCO-BUC008.00
Use Case Name:	**Receive Problem Message**
Created By:	António Mota
Creation Date:	05/01/2008
Actors:	JADE
Trigger:	This use case is initiated when the JADE cyclic behavior triggers its action (checking if there are any problem messages).
Description:	1. Save the problem received. 2. Initiate the behavior that will solve the problem.

Table 8 – Use Case "Receive Problem Message"

Table 9 describes the use case "Solve Problem".

Use Case ID:	SMACCO-BUC009.00
Use Case Name:	**Solve Problem**
Created By:	António Mota
Creation Date:	06/01/2008
Actors:	JADE
Trigger:	This use case is initiated when the JADE cyclic behavior triggers its action (checking if there are any problem messages) and a new message is received.
Description:	1. Prepare the message to be sent to the specialist agents. a. Obtain the descriptor of the specialist agents. b. Set a problem object (see figure 10) as the content of the message. 2. Send the message. 3. Initiate behavior that waits for the solutions.

Table 9 – Use Case "Solve Problem"

Table 10 describes the use case "Receive Solution Message".

Use Case ID:	SMACCO-BUC010.00
Use Case Name:	**Receive Solution Message**
Created By:	António Mota
Creation Date:	05/01/2008
Actors:	JADE
Trigger:	This use case is initiated when the JADE cyclic behavior triggers its action (checking if there are any solution messages).
Description:	1. Save the solution received. 2. Wait until the solutions from all the specialist agents are received. 3. Chose the best.

Table 10 – Use Case "Receive Solution Message"

Table 11 describes the use case "Show Solution".

Use Case ID:	SMACCO-BUC011.00
Use Case Name:	**Show Solution**
Created By:	António Mota
Creation Date:	08/01/2008
Actors:	-
Trigger:	This use case is initiated when the JADE cyclic behavior triggers its action (checking if there are any solution messages), and all the solutions messages were received.
Description:	1. Refresh the interface so that the chosen solution can be shown to the OCC supervisor.

Table 11 – Use Case "Show Solution"

Figure 16 presents the methods that are used in the resolution subsystem.

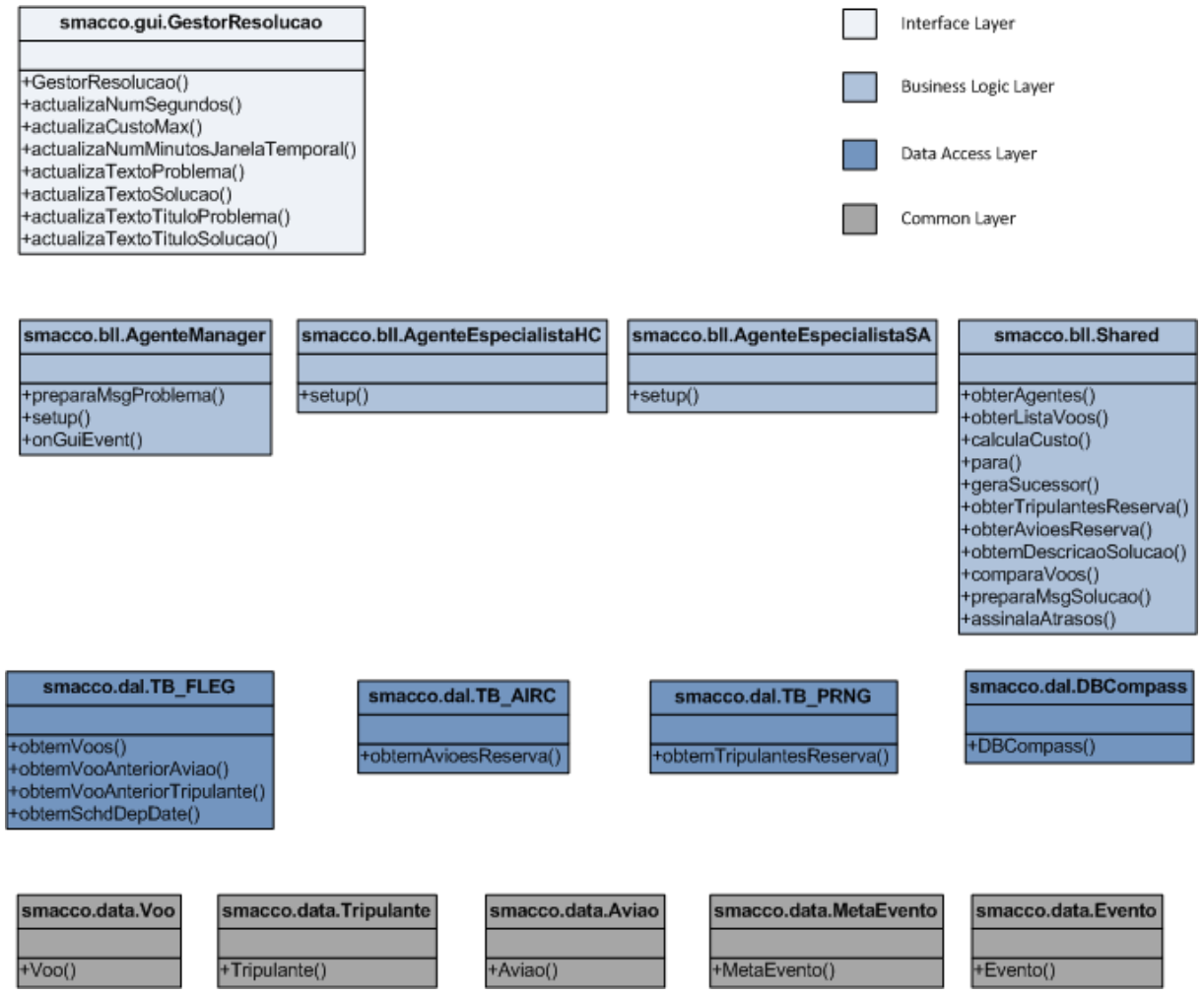

Figure 16 - Methods Used in the Resolution Subsystem

The resolution manager agent implements five behaviors:

- A *CyclicBehaviour* that waits for a problem message to arrive, using a *ReceiverBehaviour* to filter the appropriate messages.
- A *CyclicBehaviour* that waits for a solution message to arrive, using a *ReceiverBehaviour* to filter the appropriate messages.
- A *ParallelBehaviour* that execute the former behaviors concurrently.
- An *SimpleBehaviour* that sends a message with the problem to be solved to the specialist agents.

Each specialist agent implements

- A *CyclicBehaviour* that waits for a problem message to arrive, using a *ReceiverBehaviour* to filter the appropriate messages.
- A *ParallelBehaviour* that execute the former behaviors concurrently.

Two specialists agents were implemented, each one solving the problem via different meta-heuristics: hill-climbing and simulated annealing.

The hill climbing agent solves the problem iteratively by following the steps:

1. Obtains the flights that are in the time window of the problem. This time window starts at the flight date, and ends at a customizable period in the future. This will be the initial solution of the problem. The crew members and aircraft exchanges are made between flights that are inside the time window of the problem.
2. While some specific and customizable time has not yet passed, or a solution below a specific and customizable cost has not been found, repeats steps 3 and 4.
3. Generates the successor of the initial solution (the way a successor is generated is described in section 4.9).
4. Evaluates the cost of the solution. If it is smaller than the cost of the current solution, accepts the generated solution as the new current solution. Otherwise, discards the generated solution. The way a solution is evaluated is described in section 4.9.
5. Send the current solution to the agent that manages the resolution process.

The simulated annealing agent solves the problem iteratively by following the steps:

1. Obtains the flights that are in the time window of the problem.
2. While some specific and customizable time has not yet passed, or a solution below a specific and customizable cost has not been found, repeats steps 3 and 4.

3. Generates the successor of the initial solution (the way a successor is generated is described in section 4.9).

4. Evaluates the cost of the solution. If it is smaller than the cost of the current solution, accepts the generated solution as the new current solution. Otherwise, it accepts the generated solution with a probability that is given by

$$P = \exp(-\Delta E \div T)$$

where ΔE is the difference between the cost of the generated solution and the cost of the current one, and T is given by

$$T_{t+1} = \alpha T_t$$

where α lies between 0 and 1.

α was given an value of 0.8, and T was given an initial value of 10.
T is updated every N iterations. N was given a value of 2.

The way a solution is evaluated is described in section 4.9.

5. Send the current solution to the agent that manages the resolution process.

4.8 Resolution Subsystem Implementation

Figure 17 shows the implementation of the hill-climbing algorithm on one of the specialist agents.

```
while(!Shared.para(problema.getNumSegundos(), segundosExecucao, problema.getCustoMaximo(), custoSolucaoActual))
{
    // obter sucessor
    sucessor = Shared.geraSucessor(Shared.copiaArrayList(solucaoActual));

    // verifica se o sucessor tem um custoSolucaoActual menor que a solucao actual
    custoSucessor = Shared.calculaCusto(sucessor, solucaoInicialPlana);
    System.out.println("Custo Sucessor: " + custoSucessor + "\n");
    if(custoSucessor < custoSolucaoActual)
    {
        solucaoActual = sucessor;
        custoSolucaoActual = custoSucessor;
    }
```

Figure 17 – Hill-climbing Algorithm Implementation

4.9 Solution Generation and Evaluation

The generation of a new solution is made by finding a successor that distances itself to the current solution by one unit, that is, the successor is obtained by one, and only one, of the following operations:

- Swap two aircrafts between flights that belong to the flights that are in the time window of the problem.
- Swap two crew members between flights that belong to the flights that are in the time window of the problem.
- Swap an aircraft that belongs to the flights that are in the time windows of the problem with an aircraft that that is not being used.
- Swap a crew member of a flight that belongs to the flights that are in the time window of the problem with a crew member that isn't on duty, but is on standby.

When choosing the first element (crew member or aircraft) to swap, there are two possibilities:

- Choose randomly.
- Choose an element that is delayed.

This choice is made based on the probability of choosing an element that is late, which was given a value of 0.9, so that the algorithms can proceed faster to good solutions (exchanges are highly penalized, so choosing an element that is not late probably won't reduce the cost, as a possible saving by choosing a less costly element probably won't compensate the penalization associated with the exchange).

If the decision is to exchange an element that is delayed, the list of flights will be examined and the first delayed element is chosen. If the decision is to choose randomly, then a random flight is picked, and a crew member or the aircraft is chosen, depending on the probability of choosing a crew member, which was given a value of 0.85.

When choosing the second element that is going to swap with the first, there are two possibilities:

- Swap between elements of flights
- Swap between an element of a flight and an element that isn't on duty.

This choice is made based on the probability of choosing a swap between elements of flights, which was given a value of 0.5.

The evaluation of the solution is done by an objective function that measures four types of costs:

- The costs with crew members. Those costs take into consideration the amount that has to be paid to the crew member (depends on the duration of the flight), and the base of the crew member (for instance, assign a crew member from Oporto to a flight departing from Lisbon has an associated cost that would not be present if the crew member's base was Lisbon).

- The costs with aircrafts. Those costs take into consideration the amount that has to be spent on the aircraft (depends on the duration of the flight), and the base where the flight actually is.

- The penalization for exchanging elements.

- The penalization for delayed elements. The cost associated with this aspect is the highest, because the goal is to have no delayed elements.

These types of costs are taken into account in the following formula:

$$\boldsymbol{CrewMeanCost + AircMeanCost + ExWeight \times NumExchanges + DelayWeight \times NumDelays}$$

Where,

$$CrewMeanCost = \sum(CrewMemberCost * BaseCrewMemberFactor)/NumberOfCrewMembers$$

$$AircMeanCost = \sum(AircraftCost * BaseAircraftFactor)/NumberOfAircrafts$$

ExWeight was given a value of 1000, and *DelayWeight* a value of 20000. The *BaseCrewMemberFactor* and *BaseAircraftFactor* depend on the base; *BaseCrewMemberFactor* was given the values presented in table 12 and *BaseAircraftFactor* was given the values presented in table 13.

Flight / Aircraft	LIS	OPO	FNC
LIS	1	1.7	2
OPO	1.7	1	2
FNC	2	2	1

Table 12 – *BaseAircraftFactor* Values

Flight / Aircraft	LIS	OPO	FNC
LIS	1	1.3	1.5
OPO	1.3	1	1.5
FNC	1.5	1.5	1

Table 13 – *BaseCrewMemberFactor* Values

4.10 Communication Between Subsystems

The communication between the various subsystems is made through the communication between agents, using ACL messages. The agents exchange serializable objects between them (some of the data structures presented in figure 10).

Figure 18 shows the performatives of each type of message and the objects that those messages carry.

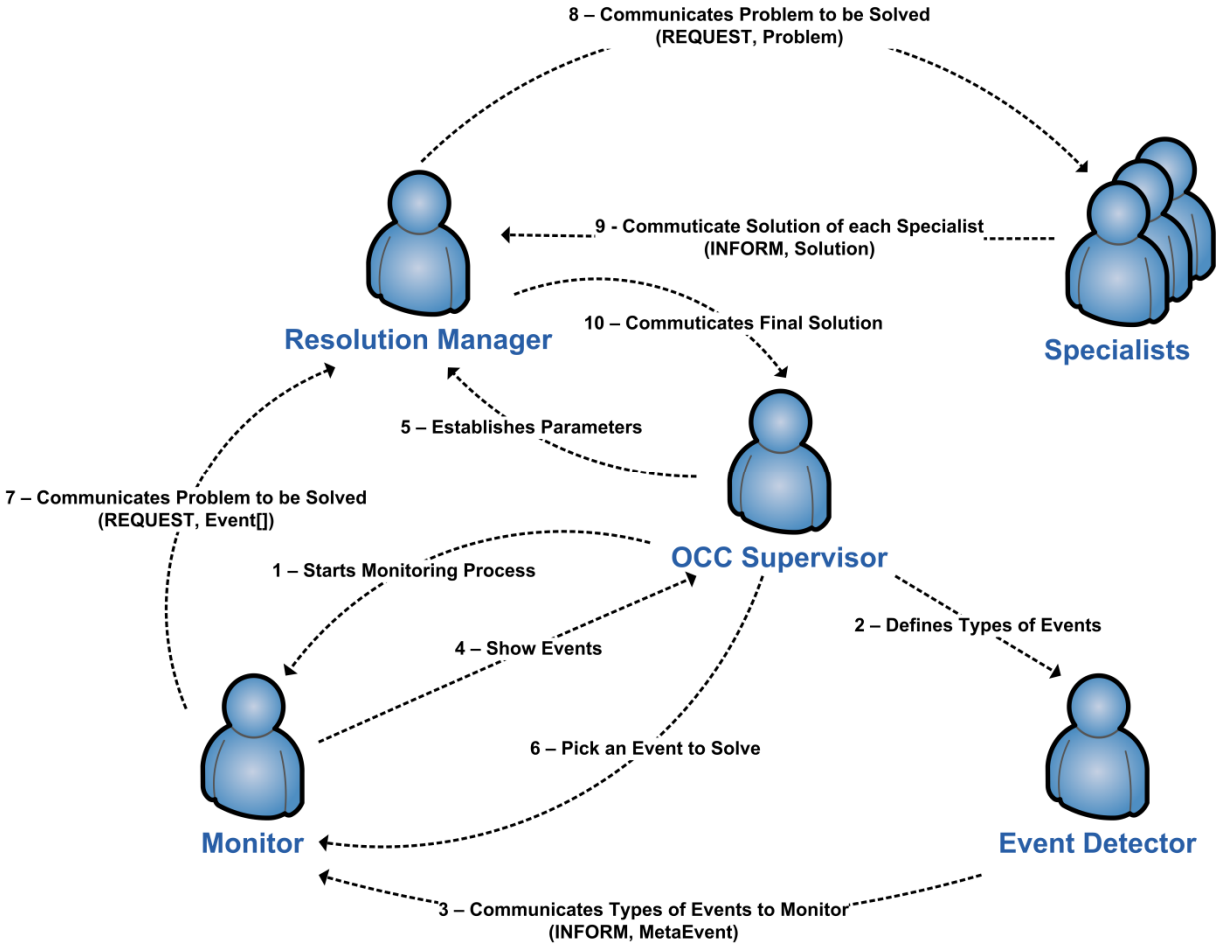

Figure 18 – Performatives and Objects of Exchanged Messages

The communication of the types of events between the event detector and the monitor is made using Inform messages that carry a MetaEvent object.

The communication of the problem to be solved between the monitor and the resolution manager is made using Request messages that carry a list of Event objects.

The communication of the problem to be solved between the resolution manager and the specialists is made using Request messages that carry a Problem object.

The communication of the solution between the specialist agents and the resolution manager is made using Inform messages that carry a Solution object.

4.11 Summary

This chapter started with the description of the overall architecture of the system, where the four types of agents (monitor, event detector, resolution manager, and specialist) were analyzed, as well as the interaction of those agents with a human entity (the OCC supervisor). All the use cases of the system were shown.

The messages exchanged between the agents transport serialized objects; the classes which they are instances of were also presented.

Then, the specification and implementation of each subsystem was described.

The chapter ended with the description of the way a solution is generated (the successor of the optimization problems described in chapter 2) and evaluated, and with the depiction of the types of messages exchanged between agents.

5 Results and Analysis

This chapter presents a real scenario, and illustrates the way the system works. First, the context of the scenario will be described, and then the behavior of the system in each phase of the disruption management process will be shown.

5.1 Delay of Flight TP428

This scenario is based on the delay of flight TP428 at 8/4/2008 17:20:00. Table 14 shows information about the flight.

Flight Number:	TP428
From:	LIS
To:	CDG
Scheduled Departure Date:	8/4/2008 17:05:00
Scheduled Arrival Date:	8/4/2008 18:41:00
Business Seats Sold:	15
Economic Seats Sold:	113
Aircraft Tale Number:	CSTTP

Table 14 - Information about TP428

Table 15 presents the crew members of the flight[1].

Crew Number	Rank	Base	Sign On	Sign Off	Cost	Perdiem
24899.7	CPT	LIS	8/4/2008 16:27:00 (actual)	10/4/2008 13:15:00 (estimated)	950	106
28267.8	OPT	LIS	8/4/2008 16:35:00 (estimated)	10/4/2008 13:15:00 (estimated)	770	203
18080.2	CCB	LIS	8/4/2008 16:24:00 (actual)	10/4/2008 23:45:00 (estimated)	670	109
21563.2	CAB	LIS	8/4/2008 16:35:00 (estimated)	10/4/2008 23:45:00 (estimated)	540	98
22330.5	CAB	LIS	8/4/2008 16:38:00 (actual)	10/4/2008 23:45:00 (estimated)	400	89
23354.4	CAB	LIS	8/4/2008 16:35:00 (estimated)	10/4/2008 23:45:00 (estimated)	340	76

Table 15 – Crew members of flight TP428

[1] Because of privacy and space reasons, only crucial information to the understanding of the problem is presented. The values for the cost and perdiem are not real.

5.2 Monitoring

Figure 19 shows the graphical user interface of the monitoring system.

Figure 19 – Monitoring System

At the upper half of the interface, the flights that are arriving (in this case, is the next 80 minutes), that already arrived (in this case, in the next 60 minutes), that are departing (in this case, in the next 90 minutes), and that already have departed (in this case, until 30 minutes ago) are shown. The flight TP480 is, as expected, in the group of the flights that will depart.

At the bottom half of the interface, details are shown about the flight, its crew members, and the aircraft used.

The monitoring system is constantly monitoring the operation, so that the information presented in the interface is maintained actual.

In this case, the system was not detecting any event, because the event system didn't have any type of event registered to be detected.

5.3 Event Detection

Figure 20 shows the graphical interface of the event detection system.

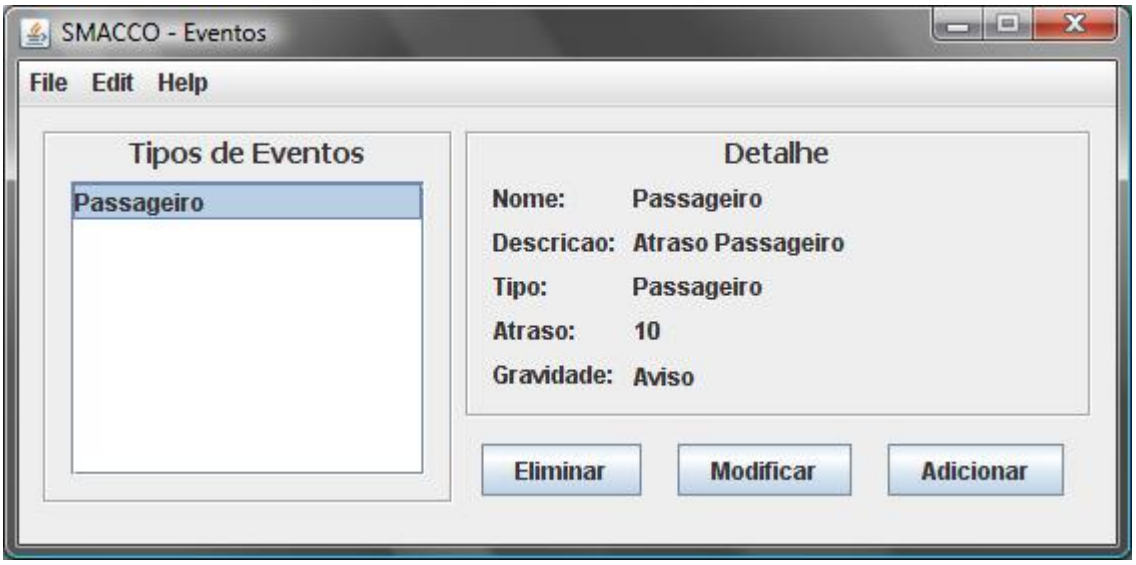

Figure 20 – Event Detection System

Through this interface, the OCC personnel are able to manage the type of events that must be monitored.

Figure 21 shows the addition of a type of event.

Figure 21 – Event Type Addition

In this particular case, the OCC personnel defined the following types of events:

- **Type of Event A**: A 10 minutes delay of a passenger is an event that should trigger a warning in that flight.

- **Type of Event B**: A 10 minutes delay of an aircraft is an event that should trigger a problem.

- **Type of Event C**: A 10 minutes delay of a crew member is an event that should trigger a problem.

After the definition of the types of events to be monitored, the event detection system cooperates with the monitoring system.

Figure 22 shows the result of that cooperation.

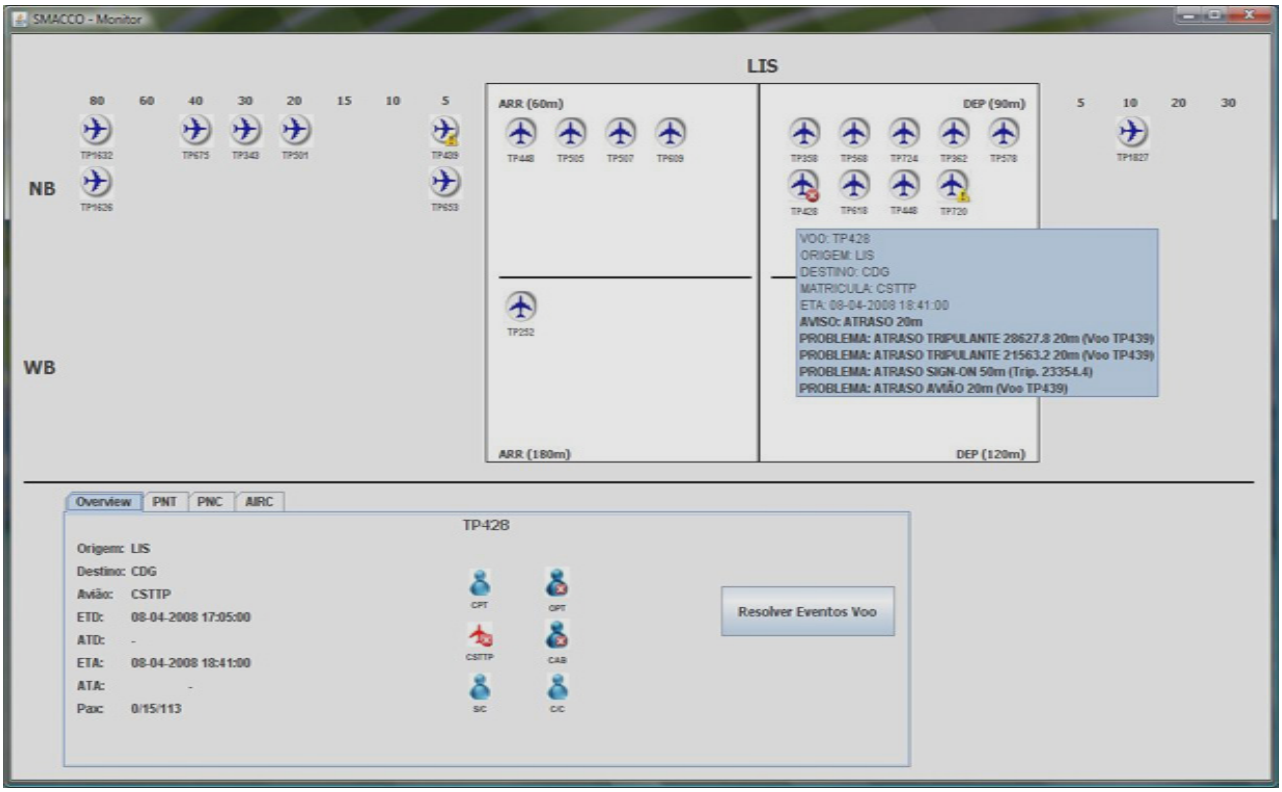

Figure 22 – Integration of the Monitoring System with Event Detection System

As can be seen at the figure, the system shows two warnings and one problem.

Before describing those warnings and that problem, it's useful to present information about flight TP439, because this flight, as it will be seen, was a cause for some of the problems of flight TP428.

Table 16 shows information about the flight.

Flight Number:	TP439
From:	ORY
To:	LIS
Scheduled Departure Date:	8/4/2008 14:10:00
Scheduled Arrival Date:	8/4/2008 16:30:00
Business Seats Sold:	15
Economic Seats Sold:	113
Aircraft Tale Number:	CSTTP

Table 16 - Information about TP439

Back to the warnings and problems detected, the flight TP439 shows a warning as a result of Type of Event A, because the flight should have arrived more than 10 minutes ago and, as a consequence, all the passengers are more than 10 minutes late.

The same happens with flight TP720, that should have departed more than 10 minutes ago, which again makes all the passengers in it behind schedule.

Finally, according to the system, the operation of the airline company has one warning and four problems on flight TP428:

- The flight is 20 minutes late and, therefore, that triggers Type of Event A, and a warning is shown.

- Crew members 21563.2 and 28627.7 are more than 10 minutes late, and that triggers Type of Event C, and two problems are shown. That crew members are late because they are on flight TP439, which is late.

- Crew member 23354.4 is more than 10 minutes late, and that triggers Type of Event C, and a problem is shown. That crew member did not report for duty.

- Aircraft CSTTP is more than 10 minutes late, and that triggers Type of Event B, and a problem is shown. The aircraft is late because it is the aircraft of flight TP439, which is late.

The OCC personnel are now able to solve the problem of flight TP438 in cooperation with the problem resolution system.

5.4 Problem Resolution

Figure 23 shows the interface of the problem resolution system, where the OCC personnel can define some parameters that will guide the resolution process, and check the solution proposed by the system.

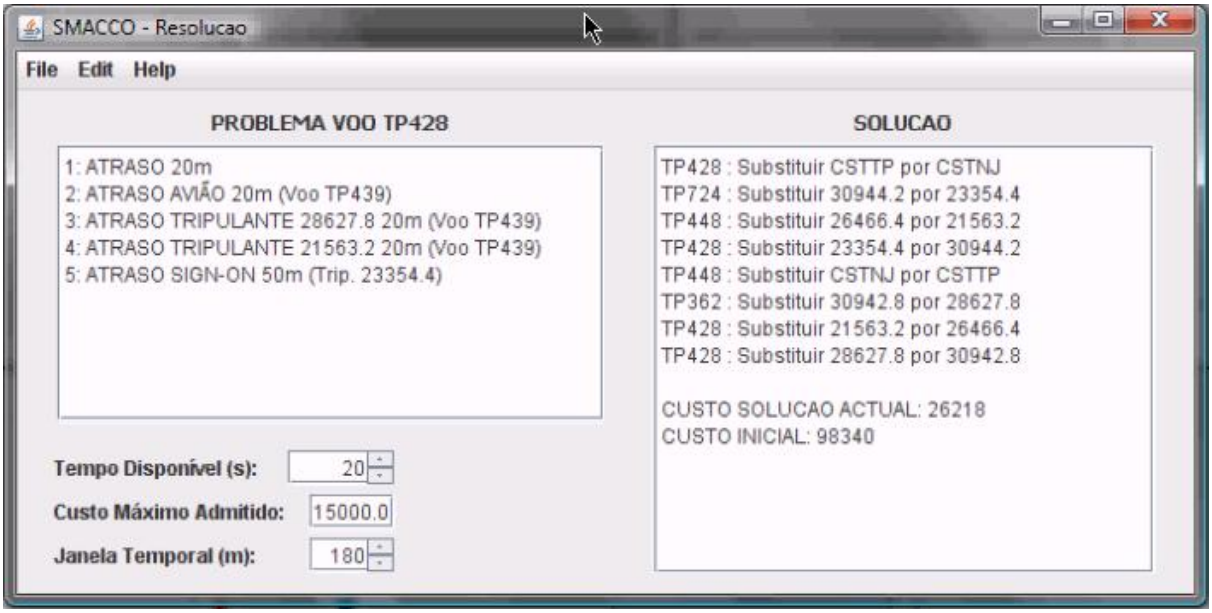

Figure 23 – Problem Resolution System

The left part of the interface shows the description of the problem, and some of the parameters that can be adjusted by the OCC personnel.

On the right side of the interface, the solution to the problem is shown. The resolution process is transparent to the OCC personnel, as they don't know how many agents participated on the process, and what methods did they use. The management of the process is the responsibility of an agent, which chooses the best solution among the ones given, and presents that solution to the OCC personnel.

In this case, the solution was:

- Switch aircrafts CSTTP and CSTNJ between flight number TP428 and flight number TP448. Because TP448 will depart later than TP428, the aircraft CSTTP has good chances of arriving before the flight's schedule departure date.

- Switch crew members 28627.8 and 30942.8 between flight number TP428 and flight number TP362.

- Switch crew members 21563.2 and 26466.4 between flight number TP428 and flight number TP448.

- Switch crew members 23354.4 and 30944.2 between flight number TP428 and flight number TP724.

As it can be seen in the figure, the solution to the problem has a lower cost than to do nothing, especially because of the high penalizations associated with crew and aircraft delays.

5.5 Summary

This chapter presented a real scenario and the way the system was used to detect the problem and solve it. In this case, the solution was to swap one aircraft and three crew members between flights.

6 Conclusions and Future Work

With the end of the nuclear part of this document, it is important to summarize each of the chapters that compose it. Firstly, a brief description about the problem that this project addressed was made, as well as the explanation of the motivations associated with its development and the establishment of its objectives and expected results. Therefore, the first chapter justified the need of this project, and the value that its implementation could bring to the airline company where it was developed.

Then, a detailed analysis of the disruption management process was made. This analysis was vital to the success of the disruption management process automation. In the second chapter, the main problem was also broken down into sub-problems, making its resolution easier, more manageable, and more effective. It was concluded that the airline disruption management process is usually a highly manual process with great potential to be automated.

The third chapter gave a description of the state of the art of the areas where the project lies and of the technologies it uses, providing a detailed view of both the scientific and the technological foundations on which this project is based. The topics analyzed ranged from optimization and metaheuristics to agents and multi-agent systems. It was concluded that the use of autonomous agents inside a multi-agent framework, that use optimization methodologies based on simple meta-heuristics, could be a promising way of solving the airline disruption management problem.

With a detailed concept of the problem to be addressed, of the objectives to be achieved, of the scientific knowledge to be used, and the tools to be utilized, it was described in the fourth chapter the work done, from the specification of the solution to its implementation. It was chosen to give more focus to the scientific component of the work, rather than to the implementation aspects. The system developed is a multi-agent system where four types of agents cooperate: monitor, event detector, resolution manager, and specialist. At the moment, there are two specialist agents: one that uses the metaheuristic steepest hill-climbing, and the other that uses simulated annealing.

Finally, the core of the document ended with the analysis of the results obtained with the deployment of the system, and for this purpose a case study was presented, so that the way the system works could be seen.

It is possible to conclude that the goals established for this project have been achieved, and that the basis for its possible near future use in the airline company was gotten underway. Indeed, it is an integrated system that automates much of the disruption management process, from the monitoring of the operation of the company in its several bases, to the detection of events and the resolution of the problems encountered.

Regarding the more scientific part of the project, it is possible to conclude that the algorithms used to search for solutions proved to be effective, having the ability to search in a clever way the vast space of existing solutions.

At last, but not least, this is a project oriented to the future: its distributed nature and the fact that it is based on agents that are specialists in solving problems easily allows the insertion into the system of new agents that solve new kinds of problems that were identified in the

meantime, or that resolve the current types of problems using different methods. It is thus a truly scalable solution, prepared to sustain the growth of the airline company.

Of course, some difficulties arose during the development of the project.

Obtaining a graphical interface that would allow the OCC personnel to view, in an easy and efficient way, the operation of the company at a particular base was a task that involved a lot of effort, with the need to refine it over time. Cooperation with some elements of the airline company was crucial in this process.

With regard to obtaining the information, the analysis of the database that contains information on the operation of the airline was an arduous task, given its complexity and the strong presence of specific language to the domain of aviation.

Finally, some difficulties have arisen in the fine-tuning of the objective function of the various meta-heuristics used, particularly in the choice of the weights to be given to each factor. Nonetheless, the number of experiments performed enabled reaching values that generated quite acceptable solutions.

Although the goals have been achieved, it is important to consider a number of improvements that could be made on future developments of this project, and that could enrich it.

In terms of the algorithms used to solve the problems, in the future other meta-heuristics can be implemented, as well as methods based in the area of operational research. The fact that this is a distributed system means that there is no theoretical limit to the number of agents that try to solve, at the same time, the same problem.

Another useful improvement for the system would be to provide it with the ability to implement the solution, which currently is made by the supervisor of the OCC. Despite being somewhat technologically accessible, this add-on to the system should be the subject of special care because of the critical nature of the operation.

Finally, it would be interesting to implement a module with machine learning algorithms, so that the system could learn from the events that occurred and from the solutions that were implemented. The knowledge obtained from this learning process could be used in defining the future operation of the airline, reducing the number of events, and helping on setting the proper resources necessary for the resolution of the events that do occur.

References

Abdelgahny Ahmed [et al.] A Proactive Crew Recovery Decision Support Tool for Commercial Airlines during Irregular Operations [Journal] // Annals of Operations Research. - 2004. - Vol. 127. - pp. 309-331.

Barnhart Cynthia, Belobaba Peter and Odoni Amedeo R Applications of Operations Research in the Air Transport Industry [Journal] // Transportation Science. - November 2003. - 4 : Vol. 37. - pp. 368-391.

Bellifemine Fabio [et al.] http://jade.tilab.com/doc/programmersguide.pdf [Online] // JADE Programmer's Guide. - 2007. - 2008.

Bellifemine Fabio, Caire Giovanni and Greenwood Dominic Developing Multi-Agent Systems with JADE [Book]. - [s.l.] : Wiley, 2007.

Bergenti Federico, Gleizes Marie-Pierre and Zambonelli Franco Methodologies and Software Engineering for Agent Systems: The Agent-oriented Software Engineering Handbook [Book]. - [s.l.] : Kluwer Academic Publishers, 2004. - ISBN 1402080573.

Bratu Stephane and Barnhart Cynthia Flight Operations Recovery: New Approaches Considering Passenger Recovery [Journal] // Journal of Scheduling. - June 2006. - 3 : Vol. 9. - pp. 279-298.

Caire Giovanni http://jade.tilab.com/doc/tutorials/JADEProgramming-Tutorial-for-beginners.pdf [Online] // JADE Programming Tutorial for Beginners. - 2003. - 2008.

Castro António Designing a Multi-Agent System for Monitoring and Operations Recovery for an Airline Operations Control Centre [Report]. - Porto : FEUP, 2006.

Castro Antonio JM Designing a MUlti-Agent System for Monitoring and Operations Recovery for an Airline Operations Control Centre // MsC Thesis. - Porto : University of Porto, Faculty of Engineering, March 2007. - pp. 1-133.

Clausen Jens, Larsen Allan and Larsen Jesper Disruption Management in the Airline Industry - Concepts, Models and Methods [Report] : Technology Report IMM-Technical Report-2005-01 / Informatics and Mathematical Modelling ; Technical University of Denmark. - Denmark : DTU, 2005.

Golany Boaz [et al.] An interactive goal-programming procedure for operational recovery problems [Journal] // Optimization and Engineering. - 2002. - Vol. 3. - pp. 109-127.

Hillier Frederick and Lieberman Gerald Introduction to Operations Research [Book]. - [s.l.] : McGraw-Hil, 2005.

Huhns Michael N Software Development with Objects, Agents, and Services [Conference] // Proceedings of the 3rd International Workshop on Agent-Oriented Methodologies. - Vancouver, Canada : [s.n.], 2004.

JADE Development Group http://jade.tilab.com/doc/index.html [Online] // JADE Documentation. - 2008. - 2008.

Kirkpatrick S., Gelatt C. and Vecchi M. Optimization by Simulated Annealing [Article].

Kohl Niklas [et al.] Airline Disruption Management - Perspectives, Experiences, and Outlook [Report]. - 2004. - Technology Report CRTR-0407.

Kohl Niklas and Karisch Stefan Airline Crew Rostering. Problem Types, Modeling, and Optimization [Journal] // Annals of Operations Research. - 2004. - pp. Vol. 127, pp. 223-257.

Kouvelis Panos and Yu Gang Robust Discrete Optimization and Its Applications [Book]. - Boston : Kluwer Academic Publishers, 1996.

Ladislav Lettovsky Airline Operations Recovery: An Optimization Approach // Ph.D. Thesis. - Atlanta : Georgia Institute of Technology, 1997.

Lee Loo Hay [et al.] Discrete Event Simulation Model for Airline Operations: SIMAIR [Conference] // Proceedings of the 2003 Winter Simulation Conference. - 2003. - pp. 1656-1662.

Love Michael and Sorensen Kim Disruption Management in the Airline Industry [Report]. - Lyngby : IMM, 2001.

Malucelli Andreia, Castro Antonio and Oliveira Eugenio Crew and Aircraft Recovery Through a Multi-Agent Airline Electronic Market [Conference] // Proceedings of IADIS International Conference e-Commerce 2006 / ed. Krishnamurthy Sandeep and Isaias Pedro. - Barcelona, Spain : IADIS Press, 2006. - pp. 51-58. - ISBN: 972-8924-23-2.

Martins Joao P and Morgado Ernesto Managing Flight Operations [Conference] // Proceedings of ATTIS'96. - London : [s.n.], 1996.

Michalewicz Zbigniew and Fogel David B How to Solve It: Modern Heuristics [Book]. - New York : Springer-Verlag, 2004.

Nocedal Jorge and Wright Stephen Numerical Optimization [Book]. - [s.l.] : Springer, 1999.

Rosenberger Jay M [et al.] A Stochastic Model of Airline Operations [Journal] // Transportation Science. - 2002. - 4 : Vol. 36. - pp. 357-377.

Rosenberger Jay M, Johnson Ellis L and Nemhauser George L Rerouting Aircraft for Airline Recovery [Report] / Georgia Institute of Technology. - [s.l.] : Technical Report TLI-LEC 01-04, 2001.

Russel Stuart and Norvig Peter Artificial Intelligence: A Modern Approach [Book]. - New Jersey : Prentice-Hall, 1995.

Trucco Tiziana [et al.] http://jade.tilab.com/doc/administratorsguide.pdf [Online] // JADE Administrator's Guide. - 2007. - 2008.

Weiss Gerhard Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence [Book]. - London : The MIT Press, 1999.

Wikipedia http://en.wikipedia.org/wiki/Hill-climbing [Online] // Hill-Climbing. - 2008.

Wikipedia http://en.wikipedia.org/wiki/Optimization_problem [Online] // Optimization Problem. - 2008. - 2008.

Wikipedia http://en.wikipedia.org/wiki/Simulated_annealing [Online] // Simulated Annealing. - 2007.

Wood Mark F and DeLoach Scott A An Overview of the Multi-Agent Systems Engineering Methodology [Conference] // Proceedings of the Agent-Oriented Software Engineering 1st International Workshop (AOSE-2000). - Limerick, Ireland : Springer-Verlag, 2001. - pp. 207-222.

Wooldridge Michael An Introduction to Multiagent Systems [Book]. - West Sussex : John Wiley & Sons Ltd., 1996.

Yang Xin-She Introduction to Mathematical Optimization: From Linear Programming to Metaheuristics [Book]. - [s.l.] : Cambridge International Science Publishing, 2008.

Yu Gang [et al.] A New Era for Crew Recovery at Continental Airlines [Journal] // Interfaces. - 2003. - pp. 5-22.