CHAPTER 6

Studying the Impact of the Organizational Structure on Airline Operations Control

Nuno Machado*, António J.M. Castro, Eugénio Oliveira†**

[*]MIEIC, DEI, Faculdade de Engenharia da Universidade do Porto, Porto, Portugal; [**]LIACC, Faculdade de Engenharia da Universidade do Porto, Porto, Portugal; [†]LIACC, DEI, Faculdade de Engenharia da Universidade do Porto, Porto, Portugal

6.1 Introduction

An organizational structure might be regarded as a set of entities collectively collaborating and contributing toward one common goal. The employees working on an assembly line or a rescue team are examples of organizational structures. Nowadays, with the increasing complexity of goods and services, and competing in a globalized world, organizations require tuned work systems, involving human capital interwoven with the latest technological innovations.

Evolving an established organizational structure is often daunting when it is behind the core mission of a business or when it operates uninterruptedly. In these cases, software simulations are an invaluable tool to explore new work practices, information flows or even decision-making processes. Modeling and simulating complete or small portions of critical workflows make it feasible to collect a set of metrics as well as introducing organizational transformations. Brought together, these factors allow for organizational performance assessment and evolution.

The work presented in this chapter is founded on such observations and aimed at proposing improvements to the operational control within a real airline company. To accomplish such an aim, we had to use a real airline company as case study. TAP, the major Portuguese air carrier, agreed to participate in such a project and provided useful information and data. As any simulation-based research, this study involved three main stages that will be discussed in the following sections. First, we had to unveil the entities involved in airline operations such as facilities, supporting systems, human collaborators, and their main activities. Next, we used Brahms, a multi-agent system featuring the BDI model and its own agent-oriented programming language to model and simulate the airline's empirical concepts. Finally, we collected a set of metrics, introduced organizational structure modifications, and established a quantitative comparison among the latter.

Advances in Artificial Transportation Systems and Simulation.

6.2 Background

First and foremost, we tried to get some background information or discover targeted literature about other initiatives regarding airline operations control simulation but to no avail.

Following this, to the best of our knowledge we were the first to simulate the Airline Operational Control Center (AOCC) organizational structure in order to study its impact on airline disruption handling. Because of that it is difficult to compare our approach with others. Nevertheless, in this section we would like to provide some background regarding work systems modeling and simulation, and also about AOCC organization and some work related to disruption management.

6.2.1 Work Systems Modeling and Simulation

A work system involves people engaging in activities over time. Human participants might not just interact with each other, but also with machines, tools, documents, and other artifacts (Calvin and Pava, 1984).

The activities performed often produce goods, services, or data. There are two different approaches when it comes to designing or improving systems: (1) machine-centered and (2) human-centered (Sierhuis and Clancey, 2002). The former is usually accomplished through a business process reengineering approach (Mayer et al., 1998) based on business process flow analysis focused on work products. The latter also takes into account how the people in the organization actually prefer to work (Greenbaum and Kyng, 1991). Unlike the machine-centered approach, which neglects human communication, collaboration, workspaces, problem solving and learning; the human-centered approach analyzes human activities, work processes or tasks, comprehensively and chronologically throughout the day (Clancey, 2002).

The human-centered work system design approach is based on modeling and simulating work practices: what people actually do, rather than their outcomes. This way, it is possible to understand the effects of human behavior in different places and times, details often omitted in a product-oriented task analysis. In the end, besides the traditional system workflow, the human-centered approach might also propose some work system transformations, including different tools, resources, locations, or scheduling.

Aiming at using a human-centered approach to model and simulate our organizational structures, Brahms (Sierhuis, 2001) was adopted as modeling and simulating tool. It follows a holistic approach to systems modeling. By developing formal models of people's behavior at the activity level, it is possible to determine the impact of these actions on the whole system.

Besides its own agent-oriented programming language, Brahms contains some predefined model components that make it straightforward to implement reality concepts:

- *Agent/groups*: to model the human collaborator;
- *Objects*: for the computerized systems;

- *Geographies*: used to indicate the location of facilities;
- *Activity*: to express the agent behavior;
- *Timing/workframes*: used to model activity duration.

Brahms does not provide real-time visual feedback of a running simulation. Therefore, this deficiency had to be addressed through the development of a visualization of the simulated airline.

6.2.2 Airline Operational Control Center Organization

The main role of the AOCC is to monitor the conformance of fight activity according to the previously defined schedule. The occurrence of some unexpected events might prevent operations taking place as planned, such as aircraft malfunction, crew delays, crewmembers absence, and so forth.

Following this, the AOCC is a human-decision system composed by teams of experts specialized in solving the described problems. Teams act under the supervision of an operational control manager and their goal is to restore airline operations in the minimum frame and at a minimum cost.

According to Castro (Castro, 2008), there are three main AOCC organizations:

- *Decision center*: the aircraft controllers share the same physical space. The other roles or support functions (crew control, maintenance service, etc.) are in a different physical space. In this type of Collective Organization all roles need to cooperate to achieve the common goal;
- *Integrated center*: all roles share the same physical space and are hierarchically dependent on a supervisor. For small companies we have a Simple Hierarchy Organization. For bigger companies we have a Multidimensional Hierarchy Organization. Figure 6.1 shows an example of this kind of AOCC organization;
- *Hub control center*: most of the roles are physically separated at the airports where the airline companies operate an hub. In this case, if the aircraft controller role stays physically outside the hub we have an organization called Decision Center with a hub. If both the aircraft controller and crew controller roles are physically outside the hub we have an organization called Integrated Center with a hub. The main advantage of this kind of organization is to have the roles that are related to airport operations (customer service, catering, cleaning, passengers transfer, etc.) physically closer to the operation.

As mentioned, Figure 6.1 shows the traditional Integrated Operational Control Center. As previously stated, the AOCC is composed of groups of workers, each one with its own responsibilities. They must report their activity to a Supervisor, translating a two-level hierarchical system. Figure 6.1 also represents the activity time-window of the AOCC, it starts 72–24 h before the day of operations and ends 12–24 h after.

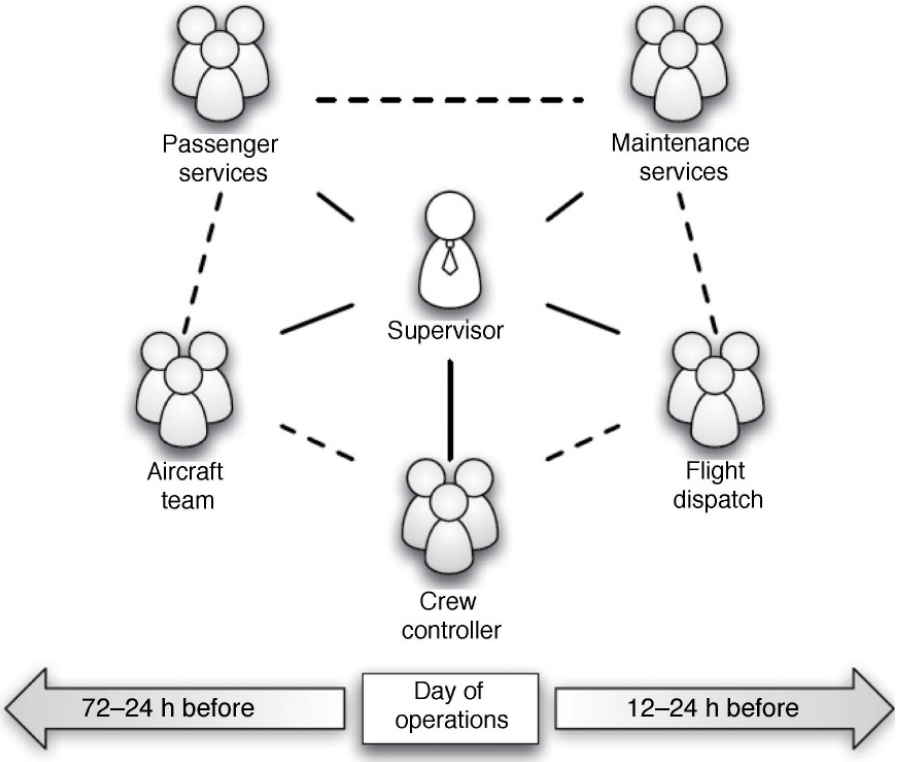

Figure 6.1: Integrated airline operational control center.
(*Source*: *Adapted from Castro and Oliveira (2010)*)

The roles most common in an AOCC are, according to Kohl and Karisch (2004) and Castro (2008):

- *Flight dispatch*: prepares the flight plans and requests new flight slots to the Air Traffic Control (ATC) entities (FAA in North America and EURO-CONTROL in Europe, for example);
- *Aircraft control*: manages the resource aircraft. It is the central coordination role in the operational control;
- *Crew control*: manages the resource crew. Monitors the crew check-in and checkout, updates and changes the crew roster according to the arisen disruptions;
- *Maintenance services*: are responsible for the unplanned services and for the short-term maintenance scheduling. Changes in aircraft rotations may impact the short-term maintenance (maintenance cannot be done at all stations);
- *Passenger services*: decisions taken by the AOCC will have an impact on passengers. The responsibility of this role is to consider and minimize the impact of the decisions on passengers. Typically this role is performed by the airports and for bigger companies is part of the HCC organization.

6.2.3 Disruption Management

Disruption Management (Kohl and Karisch, 2004), also known as Operations Recovery, is the process carried out by the AOCC when an unexpected problem prevents a flight operating as planned.

The first overview of the state-of-the-practice in operations control centers on the aftermath of irregular operations and was provided by Clarke (1998). In his study, besides an extensive review of the subject, he proposes a decision framework that addresses how airlines can reassign aircraft to scheduled flights after a disruptive situation.

Currently, the most thorough analysis of the discipline is presented by Kohl et al. (2007) where their conclusions are supported by the DESCARTES project, a large-scale airline disruption management research and development study supported by the European Union.

Other authors propose more general perspectives regarding disruption management. Yu and Qi (2004) analyze airline disruption management from different angles: crew and aircraft recovery; and applied to other fields as well: machine scheduling and supply chain coordination. Given the large scope of their work, airline operations recovery is not particularly detailed.

On the other hand, Ball et al. (2007) give insight into the infrastructure and constraints of airline operations, as well as the air traffic flow management methods and actions. Simulation and optimization models for aircraft, crew and passenger recovery are also discussed. Furthermore, the authors give an excellent survey of the airline schedule robustness as a proactive alternative to recovery, including model descriptions and a literature review.

From the mentioned studies, it is clearly a tendency to consider the disruption management problem as twofold: aircraft recovery and crew recovery. For each type of recovery several solution approaches were proposed on the basis of different methodologies.

An in-depth and comprehensive review of the most relevant studies and methodologies used in disruption management is presented by Clausen et al. (2010). They not only explain the most traditional approaches, such as Connection, Time Line, and Time Band Networks, based on the scheduled aircraft and crew rosters but also mention newer and innovative research studies.

Although the vast majority of the publications use integer programming solution methods to solve the aircraft recovery problem, the most recent works apply some metaheuristics to the problem, such as described by Andersson (2006) and Liu et al. (2006).

Moving to crew recovery, the majority of publications formulate the crew recovery problem under the assumption that the flight schedule is recovered before the crew

rescheduling decisions are made, thereby following the hierarchical structure of the disruption recovery in practice. These publications include Wei et al. (1997), Guo (2005), and Nissen and Haase (2006).

For instance, from the list of authors presented in the last paragraph, Wei et al. (1997) model the crew pairing repair problem as an integer multicommodity network flow problem on a Connection Network. The challenge is to repair the pairings that are broken and the objective is to return the entire system to the original schedule as soon as possible while minimizing the operational cost.

Something interesting about Nissen and Haase (2006) research is that it is based on European reality. They propose a duty-based formulation for the crew recovery problem, which is especially well suited for solving crew disruption for European airlines, as these, contrary to the North American airlines, employ fixed monthly crew rates, which should be taken into consideration when solving a crew disruption.

Finally, (Castro and Oliveira (2010) pioneer an approach that not only accounts for the aircraft and crew perspectives but also considers passengers. An implementation of an intelligent and distributed multi-agent system (MAS) represents the operations control center of an airline. MAS includes a crew recovery agent, an aircraft recovery agent and a passenger recovery agent. They use concepts of direct and qualitative cost to determine solutions for the disruption problem.

6.3 Empirical Airline Operations

The airline operations start way before the actual flight day as they require the scheduling of flights in advance. Then several stages emerge such as the revenue management, aircraft and crew rosters, and so on (Antonio, 2010). This is usually known as the Airline Scheduling Problem (Grosche, 2009).

When the day of operations arrives, unexpected events may prevent flights to depart as planned and the airline specialists must address those situations. This is known as the disruption management problem.

Our study is about organizational structures of the AOCC in the context of the day of operations, not to the disruption management algorithms and/or processes that are used to solve the disruptions. For that, we need to know the workflows before and after that stage of the disruption, that is, which are the unexpected events, who detects such events, how the airline specialists know about them, and who is notified of putative solutions.

In order to simulate such a scenario we needed to know the entities involved in airline operations. Figure 6.2 clearly depicts those entities and their geo-location. Squares represent facilities and ellipses computerized systems. Table 6.1 describes each of the entities' labels.

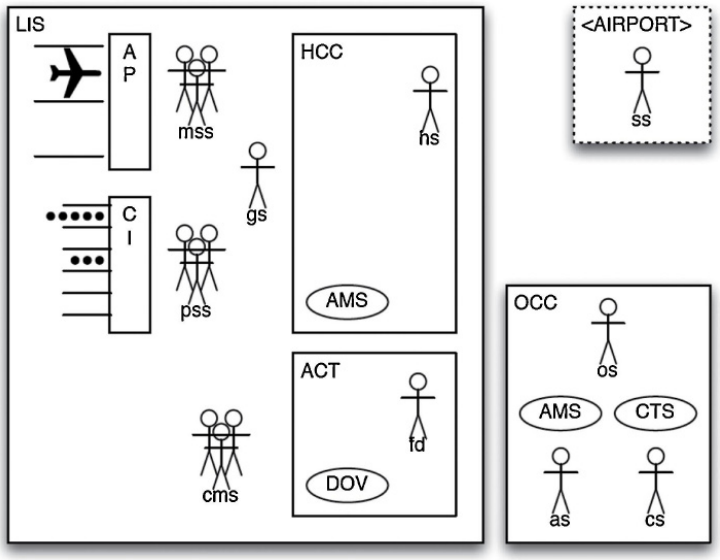

Figure 6.2: Current TAP organizational structure

Table 6.1: Organizational structure concepts.

Facilities	
ACT	Crew Terminal
AP	Aircraft Parking
CI	Passenger Check-In
HCC	Hub Control Center
LIS	Lisbon Airport
OCC	Operational Control Center
Computerized systems	
AMS	Aircraft Movement System
CTS	Crew Tracking System
DOV	Flight Operations Portal
Human collaborators	
as	Aircraft Specialist
cms	Crew Members
cs	Crew Specialist
fd	Flight Dispatcher
gs	Ground Supervisor
hs	HCC Supervisor
mss	Maintenance Services
os	OCC Supervisor
pss	Passenger Services
ss	Station Supervisor

With a big picture of the current TAP organizational structure and its components, an in-depth understanding of the workflows as well as related activities was essential. Eight workflows and activities were identified.

Concerning the activities, across the organizational structure, information is conveyed by means of VHF radios or telephones. Since the computerized systems share the same network, information is instantaneously synchronized among them and it is visible to each other. Human collaborators interact with the systems by filling forms or reading data. The specialists carry out decisions at the Operational Control Center and supervisors are required to approve those decisions.

Triggering any of the eight identified workflows is a preflight anomaly, for example, lack of fuel, aircraft malfunction, mandatory security, and so forth. If the anomaly causes a departure delay, then it is recorded on TAP databases accompanied by a delay code. Just to provide an idea of the number of potential anomalies, the proprietary delay code list of TAP has more than 200 entries, whereas the IATA international delay code list has around 80 anomaly types. An airline operator or system usually detects each anomaly, thus inquiries were made in order to classify each delay code according to concept.

In order to illustrate an operational workflow, an example follows. Imagine that 15 min before departure a ULD (Unit Load Device), inadvertently hits an aircraft during cargo loading. Assuming that this kind of anomaly has a delay code of 100 and TAP had classified such a code as being detected by the Ground Supervisor then, at this point, the Ground Supervisor is the only agent knowing about the problem. The deciding agents in the TAP organizational structure are the Aircraft and Crew Specialists, located at the OCC. They must be aware of the problem in order to find the best solution, for example, replace the aircraft, delay the flight, and so forth. Figure 6.3 illustrates the workflow behind the resolution of an aircraft anomaly detected by the Ground Supervisor.

In order to alert the Specialists, the Ground Supervisor first uses the VHF radio to communicate the problem to the HCC Supervisor. Next the latter fills a form to enter into the Aircraft Movement System and the information is propagated instantaneously to the OCC. There, the Aircraft Specialist is hopefully paying attention to the screen and becomes aware of the problem. He seeks a reason for the problem and after reaching a conclusion inputs it into the AMS, being replicated to the CTS. Now it is the turn of the Crew Specialist. Mandating or not some crew assignment changes, the Crew Specialist is required to evaluate, take action and confirm the solution suggested by the Aircraft Specialist through the CTS terminal. His input will be readily synchronized, once again, with the Aircraft Movement System, making it available to both OCC Supervisor and HCC Supervisor. As the main character in the Operational Control Center, the OCC Supervisor is required to ratify the decisions proposed by the Specialists, whereas the Hub Control Center Supervisor uses the VHF radio again to communicate changes to the Ground Supervisor.

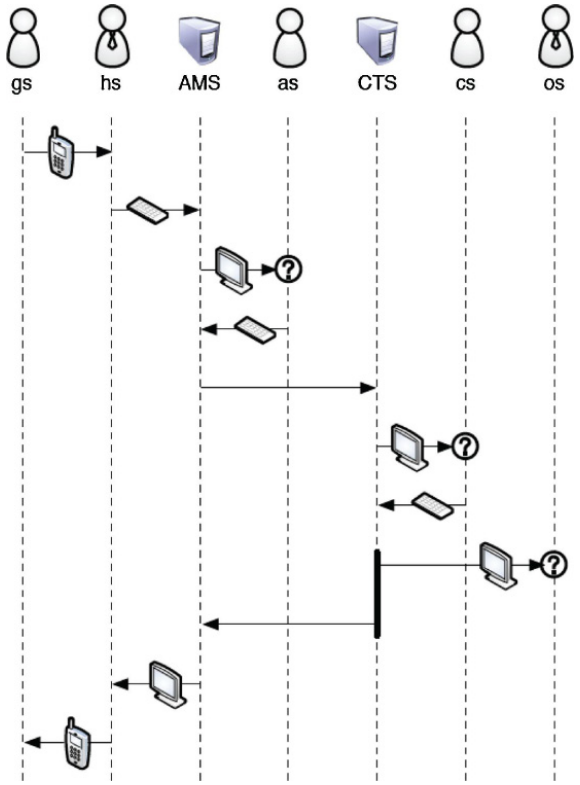

Figure 6.3: Workflow triggered by an AC anomaly detected by Ground Supervisor

All the activities above require time to perform. TAP was questioned about the duration of such activities and; although a definite answer was impossible, it provided minimum and maximum time intervals for each activity. At this point we understood that communications by phone take, on average, more time than VHF radio transmissions as they are usually concerned with more complex anomalies.

6.4 Organizational Structure Performance Assessment

This section intends to suppress the lack of information regarding the organizational structure simulation by presenting the main features of Brahms, the modeling and simulating tool that introduces a new human-centered computing paradigm. In order to accomplish this, we will first draw a big picture of the simulation system as a whole justifying the use of certain technologies and pointing out additional contributions to the communities behind those technologies. Further, we will explain how Brahms greatly improved the experience of modeling the workflows at TAP. Finally, a section will also be dedicated to expose some aspects of a visualization module developed to allow a better understanding of the concepts being simulated.

6.4.1 Background and Overall Simulation Architecture

As referred to during the introduction, the main goal of this research study was to simulate the operational control of a real airline company. Obviously, simply creating a model of such reality and mimic its intrinsic features would be of arguable interest so we aimed thereafter at proposing changes that would lead to more efficient activities and workflows.

The empirical observations listed in Section 6.3 made us aware of the reality at TAP, our case study airline company. We soon noticed that we would be treating a case that falls into the popular business-process reengineering paradigm.

Following this, we had to adopt a simulation tool that would simplify the modeling of the concepts related to our airline company while at the same time featuring some business process reengineering capabilities. Meeting these requirements was Brahms (Sierhuis, 2001) the Business Redesign Agent-Based Holistic Modeling System.

Although some theoretical information was already presented about Brahms in the Background, it is worth pointing out some technical information about this system for the purpose of clarifying certain options or side activities carried out alongside this study.

First and foremost, Brahms[1] is a Multi-Agent System featuring the BDI (Beliefs, Desires, Intentions) architecture. Whereas these characteristics are not enough to distinguish it from many other simulation engines, the Brahms Team at NASA Ames Research Center is currently developing Brahms in collaboration with the Carnegie Mellon University, and it has been successfully used in NASA's Mission Control to automate human tasks for the International Space Station. Its source code is proprietary but NASA freely distributes it for research purposes only.

At this point, we simply thought that if Brahms were enough for NASA it would certainly suit our needs. After further inspecting the features provided by Brahms, we noticed that it was a much more advanced tool than other Multi-Agent Systems that we knew about. It sports its own agent-oriented programming language, adds up some human-centered computing concepts and has its own production rules system.

The characteristics above ought to require an additional effort in implementing it in our airline company, but we decided to take the challenge. Another feature lacking in Brahms is the ability to visualize the concepts being simulated. Although this functionality was not required to get a quantitative comparison of different organizational structures, it was regarded as an educational and clarifying way of understanding the operations carried out in an airline company.

Being mostly characterized as an academic and scientific tool, Brahms lacks a wide user community, where one may get models, code examples and ask for help. In spite of that, it is thoroughly documented and their creators lead a discussion group to assist early-adopters.

[1] Software model that implements the main aspects of Michael Bratman's theory of human practical reasoning.

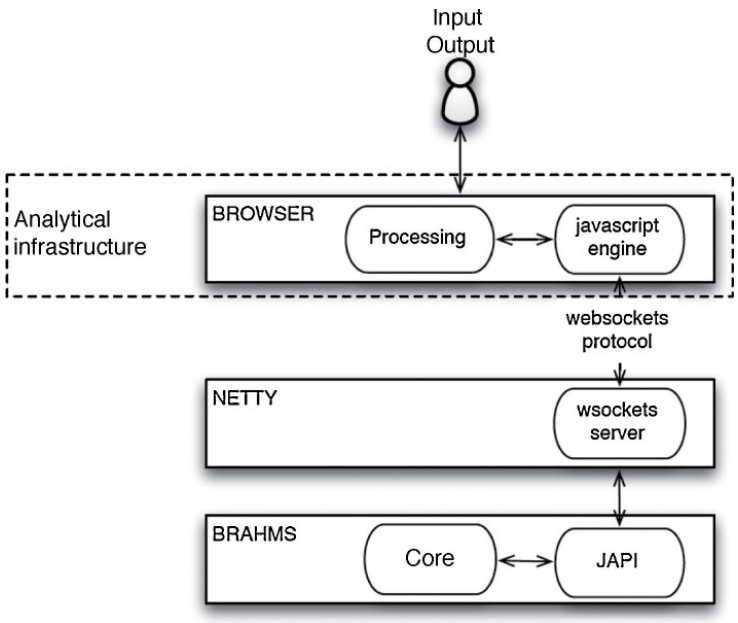

Figure 6.4: Overall simulation architecture and components

As Brahms runs in a closed virtual machine, the first contact with the simulator community was intended to inquire about the possibilities of developing a visualization of a running simulation. Although the primary approach would be to interpret a set of output files postsimulation, Brahms features a Java API (JAPI) allowing for environment expansion and control.

Interacting with JAPI would be roughly the same as interacting with a Java application. Therefore, we decided that our visualization would be built-in in the analytical infrastructure and use the latest advancements in browser technology.

Figure 6.4 depicts the components and architecture beneath the simulation portion of our study. Although this section is not meant to be too technical, other aspects of Figure 6.4 require further inspection. Starting from the beginning, the human user has the ability to interact with the analytical infrastructure through a browser. Given the set of technologies used, it is important to emphasize that at the time of writing, the only browser that supported our infrastructure was Google Chrome. Nevertheless, with fast technological evolution, it is likely in the near future other major browsers start to implement the technical innovations employed by our simulations.

Moving down in Figure 6.4, the browser portion of the simulation architecture contains the visualization module. It is mainly composed of two interwoven parts: the "javascript engine" and "processing". The former handles all the communications with the "websockets server", discussed soon, decoding the incoming messages and controlling "processing" animations.

Processing is widely used in the scientific and academic field, given its ability to create powerful representations of large sets of data. Although the original processing is based and to be

used with the Java programming language, considering our browser requirements, we had to use a javascript port of the language.

The BROWSER component also allows for simulation control, which is, starting, pausing, and stopping the simulation. The visualization module will be described in Section 4.3, so at this point it just matters understanding we are in the presence of a distributed infrastructure where messages come in, go out and an animation of the simulated theatre is displayed in-between.

The "websockets server" uses NETTY, a Java nonblocking I/O socket framework, to implement the recently introduced *websockets protocol* as part of the HTML5 specification. The use of such technology is solely implemented on Google Chrome, thus the reason our simulations only work with this browser.

The NETTY component is too technical to deserve further inspection. As any server, it establishes TCP connections with remote clients and then exchanges messages with the Java API of Brahms. Besides a message gateway, it shares some similarities with the "javascript engine" as it decodes and encodes the messages exchanged with the simulation core.

The next section will be solely concerned with the next component, BRAHMS. As we referred to, the selected simulation engine to implement our organizational structure features an agent-environment, "core" and a Java API, "JAPI". The former is more concerned with agent, geography, activities and other real entities modeling; the latter is more technical, being used for handling Java objects or other services.

6.4.2 The Simulation Module

This section is focused on the BRAHMS component of the simulation architecture (Figure 6.4). We will start by describing the "core", that is how we used Brahms formalisms and programming language to implement the empirical observation exposed in Section 6.3. As stated, the "JAPI" side is more technical, therefore we will not delve deeply into it, solely pointing out its use as a means to solve some Brahms shortcomings.

Brahms supplies a number of human-centered structural formalisms to help modeling real entities. Thus, one of the first steps in modeling a scenario with Brahms is to make a correspondence between real and artificial concepts. Table 6.2 intends to clarify our approach concerning such mapping.

Besides presenting a number of associations, Table 6.2 also hopes to illustrate the expressiveness offered by the Brahms modeling language. Although comprehensive, it just contains a subset of Brahms concepts.

The first column respects the reality, the next two contain the name of virtual entities implemented in our simulations. Starting by "Facilities and Locations", Brahms is very complete in what concerns geography modeling. There are areas and areadefs and we may not have the

Table 6.2: Concept mapping between reality and Brahms formalisms.

Reality	Brahms	
Facilities and locations	*area*	*areadef*
ACT	LisbonAirportACT	ACT (Builing)
AP	LisbonAirportAP	AP (BaseAreaDef)
CI	LisbonAirportCI	CI (Building)
HCC	LisbonAirportHCC	HCC (Building)
LIS	LisbonAirport	Airport (BaseAreaDef)
OCC	TapOCC	OCC (Building)
Computerized systems	*object*	*location*
AMS	HCC_AMS	LisbonAirportHCC
	OCC_AMS	TapOCC
CTS	OCC_CTS	TapOCC
DOV	ACT_DOV	LisbonAirportACT
(AS)	Lisbon_AS	(World)
Human collaborators	*agents*	*groups*
as	AircraftSpecialist	OCCSpecialists
cms	CrewMember	CrewMembers
cs	CrewSpecialist	OCCSpecialists
fd	FlightDispatcher	TriggeringAgents
gs	GroundSupervisor	GroundPersonnel, TriggeringAgents
hs	HccSupervisor	ApprovingAgents
mss	MaintenanceMan	GroundPersonnel, TriggeringAgents
os	OccSupervisor	ApprovingAgents
pss	PassengerMan	TriggeringAgents
ss	StationSupervisor	TriggeringAgents13

former without the latter. That is, first the area must be defined, we must specify what it is then we may name it. Looking at the "ACT" example, we first had to create a generic "ACT" extending the Building definition shipped with Brahms and then we were able to define "LisbonAirportACT" as an instance of it. In the case of "AP", Aircraft Parking, as it is not a Building, we had to choose the BaseAreaDef as parent.

Our naming conventions reveal another Brahms feature not foreseeable in the table, the "part of" construct. Listing 6.1 shows an excerpt of the "part of" and path formalisms.

Listing 6.1 Excerpt of Brahms area and path definitions

```
area   LisbonAirportAP   instanceof   AP   partof   LisbonAirport   {
}
path   LisAP_to_from_LisHCC   {
     area 1  :   LisbonAirportAP ;
     area 2  :   LisbonAirportHCC ;
     distance :     600 ;
}
```

As is clear, besides specifying what the area is, we may also specify the relation between areas (part of). Figure 6.2, shows the Aircraft Parking inside the Lisbon Airport and Brahms allows such modeling. Another very convenient feature is the path. It defines a relationship between two areas not in terms of composition but geographical dispersion; again, something very handy to set the distances (in s) between buildings or areas. The distance, on average, between the Aircraft Parking and the Hub Control Center is 10 min, so we must specify it as 600 s. As we will see later, if our agents use a vehicle, and thus only spend 2 min, the 600 s time may be overwritten in the move activity.

Moving on to the next portion of Table 6.2, it is about computerized systems. To model inanimate things, Brahms offers the concept of object. Actually, some real-world objects might be modeled as agents because, although they are physically inanimate, they may be used to reason over facts and therefore help humans take decisions, for example, a computer. The notion of agent in Brahms is a little narrower than other multi-agent systems because objects might also react and reason as agents. Apart from the naming convention used for systems, which is irrelevant, another key property is its location. In Brahms every object and agent might be given a location. Again, according to Figure 6.2, the airline systems were distributed across different locations.

Concerning the human collaborators, as stated above they were modeled as agents. As a multi-agent system, this is no surprise. The innovative factor in Brahms is the existence of groups. When implementing an agent, one may use the "memberof" keyword to set its group membership. For instance, in Table 6.2, the "AircraftSpecialist" and the "CrewSpecialist" are members of the same group, the "OCCSpecialists". This is a very powerful feature in Brahms because when we need to implement activities to be performed by the agents, we just need to implement them at the group level, then the activities are automatically inherited.

Before introducing Brahms activities, we may not skip the "Lisbon AS" object. The "AS" systems stands for Airport Screen and during our interviews with airline personnel nobody noticed its existence, thus the reason for appearing between parentheses. It is here to illustrate a simple case where a modeler needed to use a workaround to simplify or make it computationally feasible to mimic reality.

Recalling our empirical scenario, when some agents detected anomalies they would trigger workflows. In Table 6.2 they appear as members of the "TriggeringAgents" group. In reality, they perceive anomalies in the course of their activities: verifying an aircraft, checking-in passengers, loading cargo, and so on. But in our simulation we only had files with those anomalies. The closer approach would be to have those agents all reading the file and checking if they were responsible to trigger the next anomaly. Although it would be correct to do it that way, it would not be wiser because it would put too much strain on IO operations to read the same file (or check the same list), over and over again.

Following this, we created the Airport Screen system that roughly mimics those screens found at the airports with the next departures or flight delays. It reads the file and "tells"

the agents about upcoming anomalies. Now that hopefully the notion of auxiliary object was explained how about other topics worth discussion: how does the Airport Screen tell the other agents?

Answering this question definitely proves that Brahms is a fairly different multi-agent system founded on a totally new paradigm. As any other programming language, the Brahms agent-oriented programming language also supports the primitive types, such as integers, characters, etc. and the map collection. Unfortunately, it does not support lists, a major flaw that had to be overcome through the use of JAPI, a workaround explained soon. Although Brahms supports those data types, agents, and objects are unable to directly handle them. At this point is very important to underline that Brahms is human-centered and human beings do not act or reason upon integers, they do that according to facts or beliefs. This is where the BDI software model enters and somehow distinguishes Brahms from the majority of multi-agent systems.

Now that facts and beliefs were introduced, when our Airport Screen detects an anomaly it creates a fact or belief. As it is located in the "World", a Brahms abstraction to everywhere, all the agents or objects perceive such fact/belief. It is up to the modeler to implement the activities they must perform, if any, when they detect the fact.

To better illustrate the human-centered paradigm of Brahms, Listing 6.2 contains a purportedly oversimplified code excerpt. It shows a routine that every minute checks the flight list using a Java object (more about this later) and concludes a fact, triggerConcept, represented by the conceptid string. In the TriggeringAgents when the triggerConcept fact matches the conceptCode the agent does something.

Up until now we have presented an overview about how we modeled facilities, systems, and collaborators. The next step is to summarily expose the Brahms formalisms concerned with activities.

Listing 6.2 The Brahms human-centered paradigm

```
//   in   Lisbon_AS ...
repeat :   true ;
when ( knownval ( current . currMinute   >  current . cfMinute ))
do  {
string   conceptid   =  as. checkFlights ();
conclude (( current . triggerConcept   =   conceptid ),  bc :0 ,  fc :100);
}
//   in   TriggeringAgents   group ...
when ( knownval ( Lisbon_AS . triggerConcept   =   current . conceptCode ))
do  {
...
}
//   in   GroundSupervisor   agent ...
initial_facts :
( current . conceptCode   =  "gs ");
```

We identified six main activities: communicate (by radio and phone), data write, data read, reasoning, and approve. To these six, let us add a new one that will be used in our future organizational structure proposals: move between locations.

Brahms supports multiple activities, and one of them is the Java activity. The Java activity will not be discussed in detail but it is worth mentioning because it might be regarded as executing a Java method, with inputs and multiple return values. As such, it virtually allows Brahms to achieve anything possible with Java. For instance, the activity of checking flights on Listing 6.2, which required us to read from a file, could have been implemented using a Java activity.

Other types of activities, more relevant to our study were the *communicate* and the *move* activities. Concerning the former, what we knew was that certain airline operators, in the presence of an anomaly, would pick up the radio or phone and communicate such a fact to a supervisor. We also knew that such activity would consume an indefinite amount of time.

As in other systems, there are always a number of ways to implement the same scenario and our simulation was no exception. There would be several ways of communicating a disrupted flight but in our case we opted for the flight number. Ideally, it would have been better to pass a Java object because, as we will see soon, our flights were implemented as such. The problem is that agents in Brahms, as human counterparts, are solely capable of transmitting facts or beliefs, usually represented through primitive data types.

Listing 6.3 intends to show how easily Brahms makes the transmission of facts and beliefs across agents. The excerpt presented is, again, part of the TriggeringAgents group, therefore it will be inherited by multiple agents, each one with its own recipient. To surpass this issue the communicate activity shown resembles a function where the "with" field is variable.

Still on Listing 6.3 the "about" field indicates the fact or belief to be sent, in this case the disruptedFlightNumber. Once in possession of the fact or belief, the recipientAgent may act or reason upon it. Last but not least, there is the activity duration. By asserting the "random" property as true, we want Brahms to pick a value between the "min duration" and the "max duration".

Listing 6.3. The Brahms communicate activity

```
communicate   reportDisruptedFlightByPhone ( BaseGroup   recipientAgent )  {
with :   recipientAgent ;
about :   send ( current . disruptedFlightNumber );
random :   true ;
min_duration :   240;
max_duration :   480;
}
```

The way Brahms handles activity timing was of utmost importance for our study. The other activity types benefit from the same random approach and therefore the previously seen "distance" in geography paths (see Listing 6.1), may be overwritten using a move activity.

The move activity is not much different from the communicate activity, instead of a "with" and "about" properties, it has a "location" property telling the agent where to go next. The motion takes a certain amount of time that might be random, as in Listing 6.3 or static, asserting "random" as false and providing a "max duration".

Before moving to the JAPI component of the simulation architecture (refer back to Figure 6.4), a brief word goes to Brahms classes. Along with areas, objects, groups and agents, Brahms also supports classes. The problem is, these classes are not as powerful as the Java counterparts. Actually, they use the same Brahms agent-oriented programming language syntax, and the same human-centered paradigm. Therefore and simply put, classes are to objects as groups are to agents. We did not list the classes in Table 6.2 as there is solely one, the TriggeringObjects that works in a similar fashion to TriggeringAgents.

Up until now we described our approach by what concerns the modeling of the most visible concepts and activities using the Brahms proprietary agent-oriented language. Although we recognized how expressive, distinct, innovative, and somehow powerful it is, we must also underline its shallow learning curve and, as we will see further, the lack of some widely used data types and support functions.

As we stated previously, Brahms supports several primitive data types and maps. Unfortunately, lists are not available and they are one of the key data structures to store our flights. Even the flight object, which is composed of several attributes, such as scheduled departure date or flight number, would be much better abstracted by means of plain Java objects.

To address such issues Brahms provides two options. The latest alpha version allows for direct Java objects manipulation. Older versions support already mentioned Java activities. In one case or the other, there are some conventions one should respect but in the end it is roughly like calling static Java methods.

Without getting into much detail, in the implementation of our simulations, the Brahms JAPI was used in several scenarios. First and foremost, to store within ArrayLists our flights and delays objects. Second, to perform file input and output, operations not supported at the Brahms level. Third, to implement the Specialists reasoning activities. Last to stream the ongoing events to the "websockets server", see Figure 6.4.

To conclude, it is worth emphasizing that this section did not aim at thoroughly describing the implementation of our simulation using Brahms. That would require a technical manual as long as this report. The intention here was to present the human-centered nature of Brahms and how that paradigm fits the reality being modeled.

6.4.3 The Visualization Module

As a side goal, the visualization module was not required to produce the answers to the main goals of this research study. It simply receives some messages from the Brahms component, such as which activity is being carried out by which agent, and displays an animation of the simulated theatre. Therefore, the main purpose of the visualization was to provide an educational tool to allow people to learn how the airline operations management works, as well as to better understand the proposed organizational changes.

As was explained in Section 4.1, the visualization module uses Processing.js to render images and animations on the recently introduced HTML5 canvas. It is tightly connected to the javascript that decodes the messages coming from the simulation.

The visualization is composed of two distinct areas, the operational area, very similar to Figure 6.2, and an airport screen. The former is where the main action takes place, through arrows we may observe the current workflow state. The latter provides visual hints about flight departures and state. The airport screen lists all the flights within a future time frame, if a flight suffers an anomaly it is depicted in a different color and a workflow is triggered in the operational area. Assuming it was the Ground Supervisor who detected the anomaly, his next activity is to notify the HCC Supervisor, and then an arrow is displayed between him and the HCC Supervisor with a visual indicator of a radio communication.

Without being too technical, the underlying architecture of the visualization had to closely implement the concepts introduced by Brahms. This means we had to implement classes to represent the agents, the objects, the area definitions, and so on. Although requiring an additional effort, such an approach also allows for a flexible display.

Given the need to represent several distinct organizational structures, the visualization had to be dependent on the simulation. At the beginning, a list of the Brahms concepts is passed to the visualization so they can be displayed. Other features are also present such as onMouseOver actions that return further information about the concepts and so on.

To conclude, a final word goes to the amount of Processing code required to implement solely one action or even the Brahms concept. We must keep in mind that behind a Brahms concept abstraction there is a complex and large code base, therefore the need to execute the Brahms environment in a virtual machine. The problem is that we lack such constructions in Processing and we are required to implement them by hand. Following this, to implement every activity or concept is a lengthy process, the reason why the visualization will always be less expressive than the simulation itself.

6.5 Scenario and Experiments

This section aims at presenting the underlying aspects of simulation input, transformation, and output. It provides useful insights to understand the organizational results presented in the next section.

6.5.1 Simulation Input Data

As advertised, our simulations used real operational data from TAP. In the context of our research, a database service was purportedly implemented to collect pre- and postflight activity. The preoperational records included the scheduled flights, assigned aircraft, and assigned crewmembers. On the other hand, postoperational data exposed the flights that actually took off as well as aircraft and crew changes. We were also given a list with all the flights that suffered departure delays, the number of minutes, and the corresponding TAP and IATA delay codes.

Possessing such data allowed us to input the scheduled flights and treat the delays, recorded after operation, as anomalies occurring during flight handling, that is, an actual flight that suffered a delay caused by unexpected late passenger check-in, would be simulated as suffering a late passenger check-in anomaly.

It is worth emphasizing the utmost importance of using real data. In an organizational structure not all the business processes assume the same prevalence, for example, there are workflows that take place a higher number of times than others. Since we will use anomalies to trigger workflow execution, using random data would not respect the uneven distribution of processes, compromising the final results.

Our simulation was fed with the flights operated by TAP from the 15th to the 21st February of 2010, a whole seven days of activity. Although 7317 flights were scheduled to take place that week, due to data incompleteness, for example, missing databases fields, table referential deficiency, inconsistent data, we were only able to input 1801 flights, 389 of which suffered anomalies.

6.5.2 Operational Workflow Transformations

The major goal of our study was to assess distinct airline organizational structures. On the basis of the actual airline simulation, the control group, three organizational structures were incrementally changed and simulated. All the simulations were executed after the same operational scenario, comprising the scheduled flights and anomalies referred to in the previous section. When proposing organizational structure modifications we were cautious not to alter the inputs and outputs of the business process, that is, never change the triggering and deciding agents.

Our first proposal (1) suggested the removal of the HCC Supervisor. After analyzing the actual sequence diagrams, we observed that he usually plays a part as information distributor and only assumes a supervising position when facing anomalies related to Passenger Services. Removing the HCC Supervisor required three major changes in four (out of the eight) workflows. The Ground Personnel were now required to go to the Hub Control Center to input data into the AMS; OCC Supervisor accumulated the role of notifying Ground Personnel about OCC Specialists decisions; and the Passenger Services started to report anomalies to the OCC Supervisor via phone.

Proposal 2 departed from proposal 1 and aimed at avoiding the Ground Personnel to go to the Hub Control Center in order to reach the Aircraft Movement System. This way, we suggested

adding mobility support to the existing AMS, making it manageable through a wireless smartphone or laptop. Conscious of certain security implications, we decided that at this stage access would be solely granted to Ground Personnel. All the remaining operators kept interacting with AMS the same way as they did previously.

In our last proposal 3, we removed the usage restrictions on the AMS found in proposal 2 and started to think of it as a web-based system accessible from everywhere. At this stage, the Flight Dispatcher and the Station Supervisor were now able to input and read data from the AMS, no matter their location.

6.5.3 Metrics

Two metrics were used to assess organizational structure performance: overall disruption handling time and average collaborator stress. Although they are both based on the activity duration, they measure different concepts. Overall disruption handling time is the sum of the time consumed by all the workflows, that is, when an anomaly disrupts a flight it also triggers a workflow composed of several activities, which durations will be summed up until a solution for the anomaly is found. Concerning collaborator stress, it is a metric associated with each collaborator and thus requires a statistical aggregation to be used, for example, the average. It measures the number of hours spent by a collaborator in the course of a simulation.

There are activities that contribute only once to the overall disruption-handling time but several times to the collaborator stress. For instance, a phone call duration is added once to the former, but contributes twice to the overall stress, once per agent involved in the communication.

6.6 Results and Conclusion

Considering the scenarios depicted in the previous section, Figure 6.5 presents the comparison across proposals of the overall disruption handling time (left) and the average operator stress (right). The measurements are carried out in hours.

As expected, the metrics in analysis show a certain correlation, even though the collaborator stress is more affected by organizational structure transformations. The proposal that performed better was the third, achieving an improvement of 15% in the overall disruption handling and 21% for collaborator stress.

Figure 6.6 compares stress across collaborators and the proposal (chart column labels described in Table 6.1).

As one may observe, the OCC specialists (“as” and “cs”) stress remained the same across all proposals since they are deciding agents at the center of the airline workflows. In the first

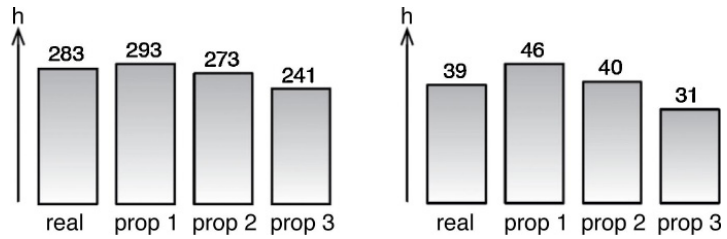

Figure 6.5: Overall disruption handling time (left) and average operator stress (right) across proposals

proposal "hs" was subtracted and "gs", "mss", and "os" suffered the highest impact. In the second, the wireless Intranet capabilities introduced in AMS, allowed the stress results to get back to the real values, except for "os". The last proposal, transform the AMS into an Internet-based system caused the highest general impact on stress.

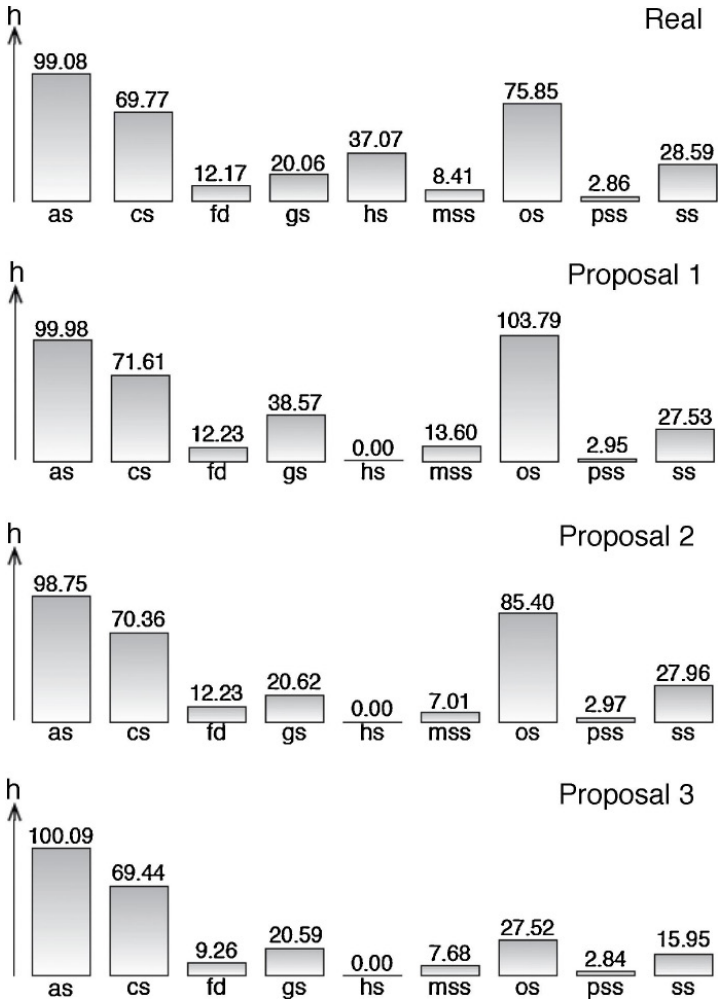

Figure 6.6: Comparison between collaborator stress and proposals

The above results proved that it is possible to assess different organizational structures according to different metrics. Beyond the analysis herein documented, the simulation of the real airline organizational structure makes it possible to evaluate other scenarios or introduce new metrics. As an abstract model based on reality, there is always room for simulation evolution.

References

Andersson, T., 2006. Solving the flight perturbation problem with meta heuristics. JoH 12 (37), 53.

Ball, M., Barnhart, C., Nemhauser, G., Odoni, A., et al. 2007. In: Barnhart, C., Laporte, G. (Eds.), Air Transportation: Irregular Operations and Control. Elsevier, Amsterdam, Handbook in OR & MS, 22.

Calvin, H.P., Pava.F P., 1984. Managing New Office Technology: An Organizational Strategy. Free Press, New York.

Castro, A., 2008. Centros de controlo operacional: Organizacao e ferramentas. Monograph for Post-graduation in Air Transport Operations. ISEC – Instituto Superior de Educação e Cîencias.

Castro, António J. M., Oliveira, Eugénio, 2010. Disruption management in airline operations control – an intelligent agent-based approach. In: Zeeshan ul-hassan Usmani PhD (Ed.), Web Intelligence and Intelligent Agents. INTECH.

Clancey, W.J., 2002. Simulating activities: relating motives, deliberation, and attentive coordination. Cogn. Syst. Rev. 3 (3), 471–499.

Clarke, M., 1998. Irregular airline operations: a review of the state-of-the-practice in airline operations control centre. J. Air Transp. Manag. 4, 67–76.

Clausen, J., Larsen, A., Larsen, J., Rezanova, N., 2010. Disruption management in the airline industry – concepts models and methods. Computers & OR 37, 809–821.

Greenbaum, J., Kyng, M., 1991. Design at Work: Cooperative Design of Computer Systems. Mahwah, New Jersey, Lawrence Erlbaum Associates.

Grosche, T., 2009. Computational Intelligence in Integrated Airline Scheduling. Heidelberg, Springer-Verlag, Berlin.

Guo, Y., 2005. A decision support framework for the airline crew schedule disruption management with strategy mapping. In: Operations Research Proceedings. Springer-Verlag, Heidelberg, Berlin.

Kohl, N., Karisch, S., 2004. Airline crew rostering: problem types, modeling, and optimization. Ann. Oper. Res. 127, 223–257.

Kohl, N., Larsen, A., Larsen, J., Ross, A., Tiourine, S., 2007. Airline disruption management: perspectives, experiences and outlook. J. Air Transp. Manag. 13, 149–162.

Liu, T.K., Jeng, C.R., Liu, Y.T., Tzeng, J.Y., 2006. Applications of multi-objective evolutionary algorithm to airline disruption management. IEEE International Conference on Systems, Man and Cybernetics, New York.

Mayer, R., Benjamin, P., Caraway, B., Painter, M., 1998. Framework and a suite of methods for business process reengineering. In: Grover, V., Kettinger, W.J. (Eds.), Business Process Change:. Idea Group Publishing, Reengineering Concepts, Methods and Technologies.

Nissen, R., Haase, K., 2006. Duty-period-based network model for crew rescheduling in European airlines. J. Sched. 9 (255), 78.

Sierhuis, M., 2001. Modeling and simulating work practice. Brahms: a multi-agent modeling and simulation language for work system analysis and design. PhD Thesis, Dept. of Social Science Informatics, University of Amsterdam, Amsterdam.

Sierhuis, M., Clancey, W., 2002. Modeling and simulating work practice: a method for work systems design. IEEE Intelligent Systems 17 (5), 32–41.

Wei, G., Yu, G., Song., M., 1997. Optimization model and algorithm for crew management during airline irregular operations. J. Combin. Optim. 1 (305), 21.

Yu, G., Qi., X., 2004. Disruption Management: Framework Models and Applications. World Scientific Publishing Company.